

AD-A080 394 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL--ETC F/6 9/2

PEDAGOG II REALIZATION.(U)

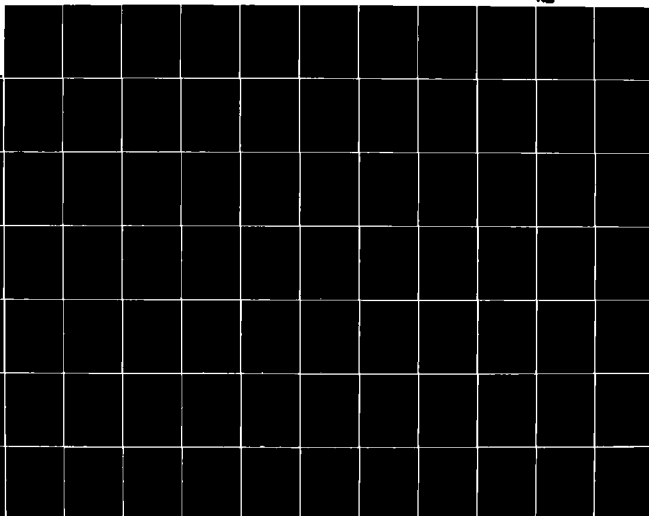
DEC 79 J S SWEEMAN

UNCLASSIFIED AFIT/CE/EE/79-088

NL

1 of 3

AD  
ADDITIONAL



AFIT/GE/EE/79-D33

①

12274

PEDAGOG II REALIZATION.

THESIS

AFIT/GE/EE/79-D33

James S. Sheehan  
Capt USAF

Approved for public release; distribution unlimited.

012225

JTB

PEDAGOG II REALIZATION

THESIS

Presented to the Faculty of the School of Engineering ✓  
of the Air Force Institute of Technology  
Air University  
in Partial Fulfillment of the  
Requirements for the Degree of  
Master of Science

by

James S. Sheehan, B.S.E.E. .  
Capt. USAF ✓

Graduate Electrical Engineering  
December 1979

12-1979  
A

Approved for public release; distribution unlimited.

## Preface

The purpose of this thesis was to complete the realization of Pedagog II, a microprogrammable display computer, intended for classroom demonstrations of microprogramming and computer register transfers.

Three previous efforts resulted in the design of Pedagog II, its control panel, and a bus interface design, all of which were non-operational at the beginning of this thesis. In addition, Pedagog II has been the object of several lab projects whose nature and exact outcome were not well documented.

The original intent of Pedagog II was to allow students to actually observe register transfers inside the machine by implementing a "step" capability at the micro level and providing the display of all major registers at the front panel. This intent has been preserved throughout the many design changes the machine has undergone and promises to make Pedagog II a very valuable and effective classroom tool.

James S. Sheehan

### Acknowledgements

The final realization of Pedagog II required many skills and talents that I could not provide. I would like to thank the many individuals who helped to make this project a success. First, I would like to thank Mr. Dan Zambon whose technical skills and enduring patience contributed greatly to the project's completion. In addition, thanks are due technicians Orville Wright, Dick Wager, and Robert Durham, who provided valuable suggestions and timely assistance.

I am very grateful to the craftsmen of the AFIT shop for fabricating the elegant cabinet and front panel of the Pedagog. The aesthetic value of the machine was enhanced many times by their artistic work.

Thanks are due to Mr. Phil Carney of the Air Force Museum who contributed his own time and efforts to the artwork of the front panel.

I am most grateful to my thesis advisor, Dr. Hartrum, and thesis committee members, Dr. Lamont and Major Ross, for their valuable criticisms and timely proddings.

Last, I would most like to thank my patient wife, whose devotion and infinite understanding allowed this project to see completion.

James S. Sheehan

## Contents

	Page
Preface . . . . .	ii
Acknowledgements . . . . .	iii
List of Figures . . . . .	vi
List of Tables . . . . .	x
Nomenclature, Abbreviations, and Terminology . . . . .	xi
Abstract . . . . .	xiii
I. Introduction . . . . .	1
Background . . . . .	1
Objective . . . . .	2
Approach . . . . .	3
Organization . . . . .	3
II. System Description . . . . .	5
Architecture . . . . .	5
Control Panel . . . . .	13
Summary . . . . .	14
III. System Analysis and Modification . . . . .	16
Switch Debounce Circuits . . . . .	16
Control State Generator . . . . .	17
Memory . . . . .	21
Run/Halt Flip-Flop . . . . .	22
Accumulator . . . . .	25
Instruction Register Decoder . . . . .	26
Microprogramming . . . . .	26
General Problems . . . . .	28
Summary . . . . .	30
IV. Control Panel and Cabinet . . . . .	31
Cabinet Layout . . . . .	31
Cooling System . . . . .	32
Power Supplies . . . . .	32
Fuses and Circuit Breakers . . . . .	33
Front Panel . . . . .	35
Panel Layout . . . . .	36
Panel Backplane Section . . . . .	39
Summary . . . . .	41

## Contents

	Page
V. Recommendations and Conclusions . . . . .	43
Recommendations . . . . .	43
Unibus/Omnibus Interface . . . . .	43
Instructions . . . . .	44
Uses . . . . .	44
Conclusion . . . . .	45
Bibliography . . . . .	47
Appendix A: Module Number Assignments and Descriptions .	48
Appendix B: Integrated Circuit Locations and Types . . .	51
Appendix C: Logic Diagrams . . . . .	70
Appendix D: Backplane Pin-to-Pin Wiring . . . . .	134
Appendix E: Microprogram Code Description . . . . .	172
Appendix F: Pedagog II Instruction Set . . . . .	176
Appendix G: Control Panel Connecting Cables . . . . .	184
Appendix H: Unibus/Omnibus Bus Interface . . . . .	186
Appendix J: Register Transfer Language Description . . .	209
Appendix K: Pedagog II User's Manual . . . . .	218
Vita . . . . .	257

## List of Figures

This thesis is a culmination of several theses. Consequently, it was necessary to retain the figure numbers from past theses that were incorporated into this work, to insure that continuity of circuit diagrams could be maintained. As a result, the figure numbers throughout this thesis are not in sequence. For ease of reference, a page number is listed after each figure.

Figure		Page
1	Signal Synthesizer (Bits 0-2) . . . . .	72
2	Signal Synthesizer (Bits 3-5) . . . . .	73
3	Signal Synthesizer (Bits 6-8) . . . . .	74
4	Signal Synthesizer (Bits 9-11) . . . . .	75
5	MPC Synthesizer and Encoder . . . . .	76
6	Buffer Driver and GP Inverter . . . . .	77
7	Multiplexer (Bits 0-3) . . . . .	78
8	Multiplexer (Bits 4-7) . . . . .	79
9	Multiplexer (Bits 8-11) . . . . .	80
10	Bus and IR Displays/Drivers . . . . .	81
11	PC and AC Displays/Drivers . . . . .	82
12	Clock Rate Selector and Lamp Test . . . . .	83
13	Unused . . . . .	
14	Unused . . . . .	
15	Omnibus Pin Assignments . . . . .	88

## List of Figures

Figure		Page
16	Bus Interface Block Diagram . . . . .	191
17	Signal Labeling . . . . .	71
18	Control Panel . . . . .	37
19	Block Diagram of Control Panel . . . . .	40
20	Voltage Converter Circuit . . . . .	84
21	Pedagog Block Diagram . . . . .	7
22	Control State Generator Timing Diagram. .	19
23	Run/Halt Flip-Flop . . . . .	85
24	Deposit Circuit . . . . .	86
25	Examine and Memory Control . . . . .	87
26	Control Switch Debounce Circuits . . . .	88
27	Bus Terminator . . . . .	89
28	Control State Generator . . . . .	90
29a	General Purpose Arithmetic Unit (M7300) .	91
29b	General Purpose Arithmetic Unit (M7301) .	92
30	Accumulator . . . . .	93
31	Accumulator Output Switch . . . . .	94
32	Accumulator Status . . . . .	95
33	Link . . . . .	96
34	Program Counter . . . . .	97
35	Program Counter Output Switch . . . . .	98
36	AC and PC Bus Control Switch. . . . .	99
37	Increment Program Counter . . . . .	100
38	Unused . . . . .	

## List of Figures

Figure		Page
39	Unused . . . . .	
40	Memory Block Diagram . . . . .	101
41	MAR Control Circuit . . . . .	102
42	Memory Address Register . . . . .	103
43	MAR Output Switch . . . . .	104
44	MAR and MMAR Output Control Switch . . .	105
45	Memory Buffer Register . . . . .	106
46	MBR Output Switch . . . . .	107
47	MBR and MMBR Output Control Switch . . .	108
48	MAR, MMAR, MBR, and MMBR Bus Control Sw.	109
49	Instruction Register . . . . .	110
50	Instruction Register Decoder . . . . .	111
51a	C-Logic . . . . .	112
51b	C-Logic . . . . .	113
52	B-Logic . . . . .	114
53a	MPC Input Signal (F4) . . . . .	115
53b	MPC Input Signal (F5) . . . . .	116
53c	MPC Input Signal (F6) . . . . .	117
53d	MPC Input Signal (F7) . . . . .	118
53e	MPC Input Signal (F8) . . . . .	119
53f	MPC Input Signal (F9) . . . . .	120
53g	MPC Input Signal (F10) . . . . .	121
53h	MPC Input Signal (F11) . . . . .	122
54	Microprogram Program Counter . . . . .	123
55	MPC Bus Control Switch . . . . .	124

## List of Figures

Figure		Page
56	Microprogram Instruction Register . . . .	125
57	MIR Decoder . . . . .	126
58a	Control Switch Field A (Codes 0-13) . . .	127
58b	Control Switch Field A (Codes 14-27) . . .	128
58c	Control Switch Field A (Codes 30-37) . . .	129
59a	Control Switch Field B (Codes 40-53) . . .	30
59b	Control Switch Field B (Codes 54-67) . . .	131
59c	Control Switch Field B (Codes 70-77) . . .	132
60	Unibus/Omnibus MD Interface. . . . .	194
61	Unibus/Omnibus MA Interface. . . . .	195
62	Unibus/Omnibus DATA Interface . . . . .	196
63	Omnibus/Unibus MD Interface . . . . .	197
64	Omnibus/Unibus DATA Interface . . . . .	198
65	Interface Control Circuits . . . . .	199
66	Switch Register . . . . .	133
67	Bus Interface IC Locations and Types . . .	201
68	Bus Interface IC Locations and Types . . .	202
69	Corrected Interface Control Circuit . . .	204
70	MRI Word Format . . . . .	240
71	RRI Word Format . . . . .	240

## List of Tables

Table		Page
I	Pedagog Input/Output Control Signals . . . .	207
II	Register Abbreviations . . . . .	220
III	Microprogramming to Emulate the PDP-8 . . .	227
IV	Microprogram Code Description . . . . .	233
V	Memory Reference Instructions . . . . .	238
VI	Register Reference Instructions . . . . .	242

## Nomenclature, Abbreviations, and Terminology

Pedagog	Pedagog and Pedagog II are used interchangeably throughout this thesis. In no case is Pedagog intended to refer to Pedagog I.
AC	Accumulator
ALU	Arithmetic Logic Unit
CPU	Central Processor Unit
DEC	Digital Equipment Corporation
Evoke	To enable or turn on
FF	Flip-Flop
GPA	General Purpose Arithmetic Unit
I/O	Input/output
IC	Integrated circuit
IR	Instruction Register
IRD	Instruction Register Decoder
L	Link, overflow bit of the accumulator
LED	Light emitting diode
LSI	Large Scale Integration
MAR	Memory Address Register
MBR	Memory Buffer Register
MIR	Microprogram Instruction Register
MIRD	Microprogram Instruction Register Decoder
MMAR	Microprogram Memory Address Register
MMBR	Microprogram Memory Buffer Register

MOS	Metal Oxide Semiconductor
MPC	Microprogram Program Counter
MSI	Medium Scale Integration
PC	Program Counter
RAM	Random Access Memory
RTM	Register Transfer Module
SSI	Small Scale Integration
TTL	Transistor-Transistor logic
Unibus	A 12-bit bus used both for addressing and data
Omnibus	A DEC bus system containing three separate busses. One is used for memory address (MA), one for memory data (MD), and one for data (DATA).

Abstract

A small microprogrammable computer, designed in a previous effort, was evaluated to determine its condition. Found non-operational, it was analyzed to determine areas that required design changes. Several modules of the system were redesigned. The system was tested to insure proper operation.

A new control panel and cabinet were designed and constructed. The computer backplane was installed in a cabinet. Existing power supplies were adapted for use, and another was designed and added.

A bus interface, designed in a previous effort to allow expansion of memory, was fabricated and tested. It was found non-operational.

The machine is microprogrammable and has a 256 word, random access memory. Microcode was written to allow the machine to emulate the 36 basic instructions of the Digital Equipment Corporation (DEC) PDP-8. All major register transfers can be seen on the front panel. The machine may be taken to the classroom for instruction or demonstrations in basic microprogramming.

## Introduction

Visible results of abstract ideas tend to reinforce the learning experience. The realization of Pedagog II into a working computer has offered the visible results that have made this project worthwhile. In the process, a working aid to computer instruction was made available for laboratory and classroom use.

## Background

The idea of a pedagogical (teaching) computer originated from an AFIT thesis by Leroy Chamberlain, Design and Simulation of a Small General Purpose Digital Computer (Ref. 4). In that thesis, a general purpose digital computer was designed and simulated on a Control Data Corporation 6600 digital computer. A following thesis, Design, Realization, and Implementation of Pedagog II (Ref. 1), written by Richard A. Beck and Richard L. Hartman, resulted in the design and construction of a machine similar in concept to Chamberlain's design, but expanded to allow microprogramming.

The control panel of the original Pedagog II was only partially realized by Beck and Hartman. A project by Hugh G. McKay and Peter G. vonGlahn, Design, Realization, and Implementation of Pedagog II control Panel (Ref. 2), resulted in the addition of an octal display and the ability to display more of the internal registers. An investigation by Winfred G. Parris, Pedagog II Enhancement and Expansion (Ref. 3), led to the design of a

Unibus/Omnibus interface to allow the 256 word random access memory to be replaced by a 4096 word core memory. The interface was designed but not fabricated. The purpose of this project is to continue those previous efforts to complete the Pedagog II.

### Objective

The objective of this project was to combine the results of previous Pedagog efforts into a working computer system for laboratory and classroom use. Specifically, the following were to be accomplished:

- Evaluate the condition of Pedagog and make any repairs or modifications necessary to make it a working system.

- Redesign and fabricate a control panel and cabinet for Pedagog which would give it classroom portability, and make it more conducive to student use.

- Complete and verify Pedagog microprogramming to allow the emulation of the 36 basic operations of the DEC PDP-8 minicomputer.

- Assemble all available information concerned with Pedagog into one source document.

In bringing all previous Pedagog efforts into one document, many references will be made throughout this thesis. This is not so much to refer the reader elsewhere as it is to give credit to the proper source. All material from previous documents believed to be pertinent to the use or maintenance of Pedagog are

incorporated, as well as any recent modifications or corrections. The bulk of past materials are presented in appendices where practical.

### Approach

This project was divided into three phases: 1) system analysis and modification, 2) panel and cabinet design and fabrication, and 3) microprogramming.

The first step of this project was one of becoming familiar with the system. After the state of the system was assessed, problem areas were determined and parts and accessories for modifications and redesigns were put on order.

The second phase of the project required the redesign of a front panel and cabinet. During phase one, it was determined that some enhancements to the panel were necessary. These additions were included in the new panel design.

Phase three required programming and testing the machine microcode. This phase was necessarily delayed until the system was operating properly. The microcode was tested by executing each of the 36 basic PDP-8 instructions and checking where each instruction mapped into memory. Then the microcode at that location was checked for proper operation.

### Organization

Chapter 2, System Description, describes the system configuration of Pedagog by defining its architecture. Chapter 3,

System Analysis, deals with the modifications and repairs necessary to make the Pedagog operate. The control panel and cabinet are described in Chapter 4. Power supply analysis and design are also discussed in Chapter 4. Chapter 5 concludes this investigation and recommends some areas for future expansion and application of the Pedagog.

Two appendices are of special note. Appendix H contains the advances made on a bus interface circuit. Appendix K is a user's manual, complete with sample programming.

## II. System Description

This chapter describes the architecture and control panel of Pedagog II. No attempt will be made at this point to describe which of the parts of the computer and control panel are, or are not, working. That will be discussed in Chapter 3. This chapter describes Pedagog on a module by module basis to the depth needed to understand later chapters. The discussion is in two sections: 1) architecture, and 2) control panel. Each module of the architecture will be described along with its function in the system. The description of the control panel will include its general design features and a brief description of their functions. Modifications and enhancements to the panel will be discussed in detail in Chapter 4.

### Architecture (Ref. 1:9-17)

The Pedagog is a 12-bit, microprogrammable computer designed specifically to emulate the DEC PDP-8 minicomputer. The Pedagog architecture uses a single bus called a Unibus. Each module of the Pedagog uses the single bus to transfer information to other modules of the system. Modules which comprise the Pedagog system are described in the following paragraphs. The modules are:

General Purpose Arithmetic Unit

Accumulator

Link

Program Counter

Memory, 256 word, scratchpad  
Memory, 4096 word, core (Proposed. See Appendix H)  
Memory Address Register  
Memory Buffer Register  
Microprogram Memory Address Register  
Microprogram Memory Buffer Register  
Microprogram Program Counter  
Instruction Register  
Instruction Register Decoder  
Microprogram Instruction Register  
Microprogram Instruction Register Decoder  
Unibus  
Unibus Terminator  
System Clock  
State Generator

Several of the modules are DEC commercial modules. They are the General Purpose Arithmetic Unit, 256 word scratchpad memory, Unibus Terminator, and System Clock. All modules associated with the 4096 word core memory are also commercial units. The relationship of each of the modules to others in the system is shown in Figure 21, page 7.

The General Purpose Arithmetic unit (GPA) is a DEC register transfer module which accomplishes the arithmetic and logical operations. It contains two sixteen-bit registers (A-register and B-register). Only the twelve lower bits are used to be

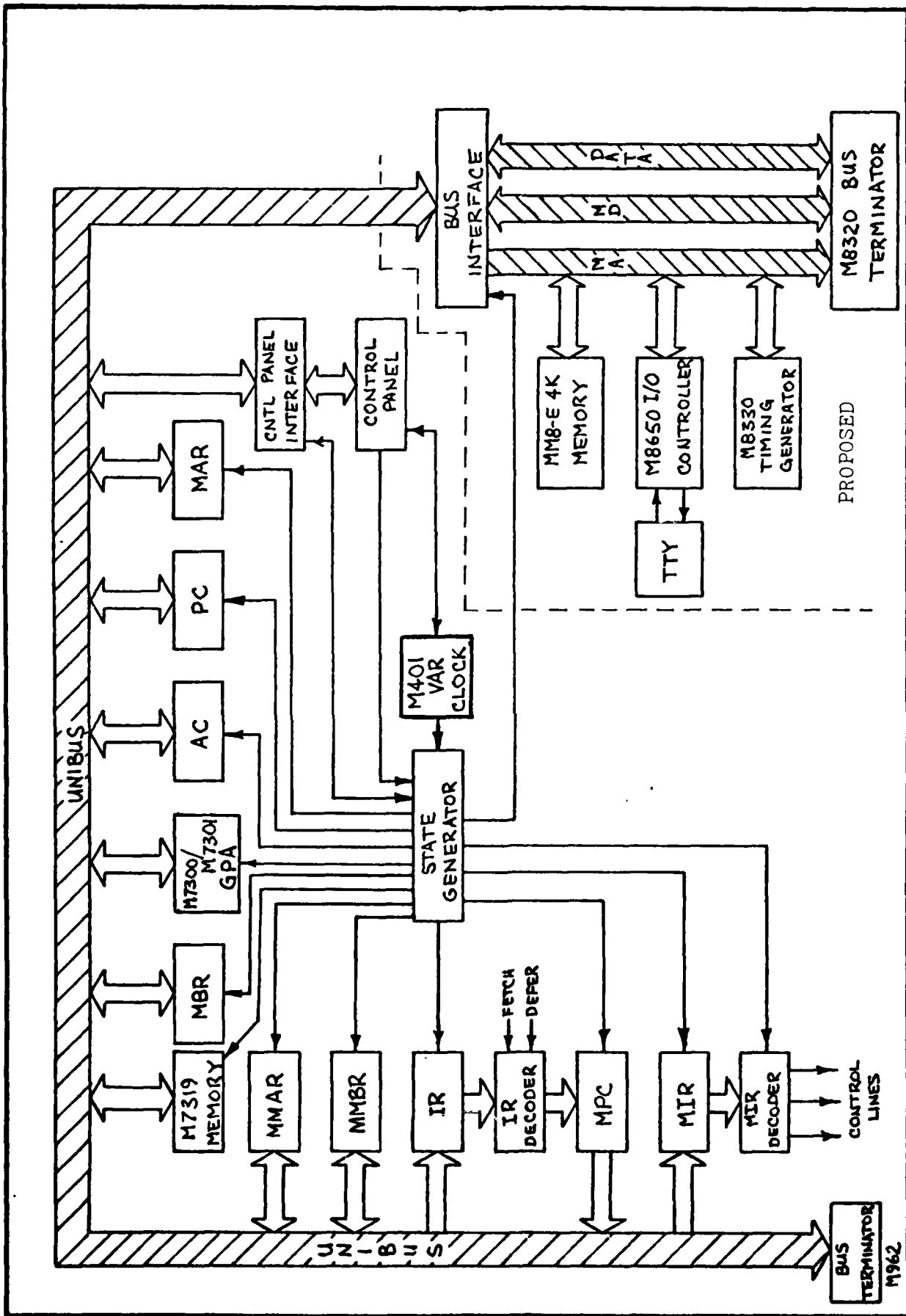


Fig. 21. Pedagog Block Diagram

consistent with the 12-bit word length of the machine. The GPA consists of two boards (M7300 and M7301). The two boards must be used together and are connected by an edge connector (H851). The GPA performs 10 arithmetic and logical operations. They are: A(not), A+1, A+B, A-B, B(not), A AND B, A OR B, A XOR B, A left shifted once, and A right shifted once. The GPA is not available to the programmer directly. It is used at the microprogram level and its results are stored in the accumulator.

The accumulator (AC) is a twelve-bit register available to the programmer under program control. It is used as an interface register between memory and the GPA. Data can be loaded into the accumulator from the bus. The data can be shifted left or right while in the accumulator. All other data manipulations are done under microprogram control in the GPA with the results transferred back to the accumulator. The accumulator contents are available at an LED display on the control panel.

A Link bit is connected to the accumulator and extends it to a thirteen-bit register. The Link can be shifted into the accumulator during a right shift operation, or the accumulator can be shifted into the Link during a left shift operation. The content of the Link is displayed on the front panel with the accumulator. Logical operations can be performed based on Link status (set or reset). This allows the programmer to check a particular accumulator bit by shifting into the Link.

The Program Counter is a twelve-bit register used to store the program address at the main program level. It can be loaded from the front panel or under program control. The contents of the Program Counter are used to set the word address in memory. The PC can accept data from the bus or load data onto the bus. It is an incrementable register.

Presently the Pedagog uses a 256 word, sixteen-bit/word scratchpad memory. This is a random access semiconductor memory (M7319) with MOS integrated circuits used for storage. Only the lower twelve bits are used with the upper four bits wired low. All control signals to the memory are under micro-program control.

A 4096 word, twelve-bit/word bulk core memory was originally specified for Pedagog. The 4096 word memory was not immediately available so the 256 word memory was substituted temporarily. The core memory and its associated circuitry are shown in the "proposed" section of Figure 21, page 7. The larger core memory has been procured and its interface to the Pedagog is discussed in Appendix H. Additional discussion is included in the recommendations section of Chapter 5.

The Memory Address Register (MAR) is a twelve-bit register used to interface between other system registers and the address section of memory. It is connected directly to the bus and can receive an input from any other register whose output is connected

to the bus. It is normally used as a buffer register between the PC and memory. The MAR is used at the main program level of execution.

The Memory Buffer Register (MBR) is a twelve-bit register used as an interface between other system registers and memory. During normal operation, all data reads into and out of memory through the MBR. The MBR is used at the main program level of execution.

The Microprogram Memory Address Register (MMAR) is a twelve-bit register and serves the same function as the MAR except at the microprogram level of execution. A separate register is required at both levels because the same memory is used at both levels. Several memory access operations are required at the microprogram level for each operation at the main program level.

The Microprogram Memory Buffer Register (MMBR) is a twelve-bit register that serves the same function as the MBR except at the microprogram level of execution. This register is required in addition to the MBR for the same reason discussed under MMAR above. The MMBR's input and output are connected to the bus. It can accept data from any bus-connected register or can serve as a source for these registers.

The Microprogram Program Counter (MPC) serves the same function as the PC but at the microprogram level. It is used as an instruction location indicator. The only difference

between MPC and PC operation is the input. Where the PC can serve as a destination register for any bus-connected register, the MPC input always originates at the instruction register decode circuit. The MPC input is not connected to the bus. During microprogram execution, the MPC is always set to the next memory location to be addressed.

The Instruction Register (IR) is a twelve-bit destination register for main program instructions. Its input is bus-connected but its output is not. All main program instructions read from memory transfer to the IR through the MBR.

The IR Decoder (IRD) is used to decode main program instructions. The twelve-bit output of the decoder is the starting address in memory which contains the first instruction of the microprogram. This is the microprogram which will perform the necessary register transfers to accomplish the program instruction. The IRD uses asynchronous logic and its output is present at the MPC input at all times.

The microprogram Instruction Register (MIR) is a twelve-bit register used as the destination register for microprogram instructions read from memory. Its input is connected to the bus and its output is connected to the Microprogram Instruction Register Decoder (MIRD).

The MIRD is used to decode the microprogram instruction and provide the control signals needed to accomplish register

transfers. The MIR provides a twelve-line input that is split into two six-bit parts in the MIRD. The remainder of the decoder then decodes the six inputs to one output. This allows two register transfer codes to be generated simultaneously. In this way the output of a source register and the input of a destination register can be enabled during one microinstruction. Each decoder half is capable of  $77_8$  outputs.

The bus utilized by Pedagog is the DEC Unibus. This single bus is used for all data, main program instructions, and microprogram instructions. This requires that the bus be shared by all modules of the system. The bus is twelve bits wide. When not in use, all twelve lines are held at a logical high (3.4 VDC) by the bus terminator. The bus is asserted low (0.0 VDC).

The bus terminator is a DEC RTM (M692). It contains a voltage divider network for each bus line, supplying the necessary 3.4 VDC.

The variable clock is a DEC RTM (M401). It serves as the master clock for the system. It is variable by external control up to 10 megahertz. A variable potentiometer on the RTM allows for adjustment to a specific value within each of five different ranges. The clock is discussed in detail in Chapter 4.

The state generator is the only synchronous portion of the Pedagog. The main purpose of the state generator is to provide the system timing for register transfers and memory references.

The state generator has undergone considerable redesign and will be discussed in detail in Chapter 3.

### Control Panel

The Pedagog control panel is used to initiate machine operation, enter data, display data, and halt machine operation. Five displays are available to view register contents. Four of the displays are made of individual light emitting diodes (LED). Each LED display uniquely displays the contents of a particular source. The four sources are the accumulator, Unibus, PC, and IR. A Link bit is mounted with the accumulator and is used to display the contents of the Link register. The fifth display (an octal display) shows the contents of one of seven possible registers (MMAR, MMBR, MAR, MBR, MPC, MIR, IR) or the Unibus. The particular register to be displayed is determined by a selector switch on the panel.

Two circuit cards in the backplane are directly associated with the panel. They are used primarily to duplicate the register contents of the MBR, MMAR, MMBR, MIR, and IR, for display on the front panel. These two modules are discussed in detail in Chapter 4.

Twelve switches on the Pedagog control panel are used as a Switch Register. The Switch Register is used to enter data or addresses into the machine from the front panel.

Seven function switches on the front panel are used to

control the operation of Pedagog. The function of the seven switches are as follows:

a. LOAD--allows the address in the Switch Register to be loaded into the Program Counter.

b. DEPOSIT--stores the contents of the Switch Register in the memory location indicated by the Program Counter.

c. EXAM--examines the memory location specified by the Program Counter. The switch must be activated twice. The first activation displays the contents of the memory location, and the second activation clears the memory contents from the bus.

d. CLEAR--sets the GPA, accumulator, and Link to a binary zero, sets the state generator to the Halt state, and sets the MPC to address 7400 in memory. Memory is also cleared of any pending read/write operation.

e. RUN--activates the system to full operation. The Pedagog will run only from the Halt state.

f. HALT--stops system operation. The Pedagog can also be halted under software control by using the HALT command (see Appendix F).

### Summary

This chapter has described the architecture and control panel of Pedagog II. The machine contains a general purpose arithmetic unit to perform logical and arithmetic functions. The memory unit serves as a storage for both microprograms and main programs.

The registers included in the design are used as address registers, buffer registers, and storage registers, and serve mainly to aid the GPA and memory.

The machine is microprogrammable. This allows changing the operation of the machine at the register transfer level by changing the microprogramming.

The Pedagog control panel has five displays to allow observation of internal register transfers. The panel also contains a 12-bit switch register to input data and addresses, and seven function switches for control.

The Pedagog required many design modifications in order to operate properly. The next chapter describes these modifications and explains why they had to be made.

### III. System Analysis and Modification

When this project was begun, Pedagog II was not an operational computer. It was believed by Parris (Ref. 3) that only the memory prevented the machine from operating successfully. However, since the memory was not usable, complete verification of other Pedagog circuits could not be made. This chapter deals with the step-by-step analysis of Pedagog deficiencies, both those documented in former projects and those discovered during this analysis. Typically, as each problem was discovered and corrected, the machine operation improved enough to detect other errors. Using this technique, eight major problems were discovered. As each problem area is addressed, the particular effects of that problem are discussed and the actions taken to alleviate the problem are presented.

#### Switch Debounce Circuits

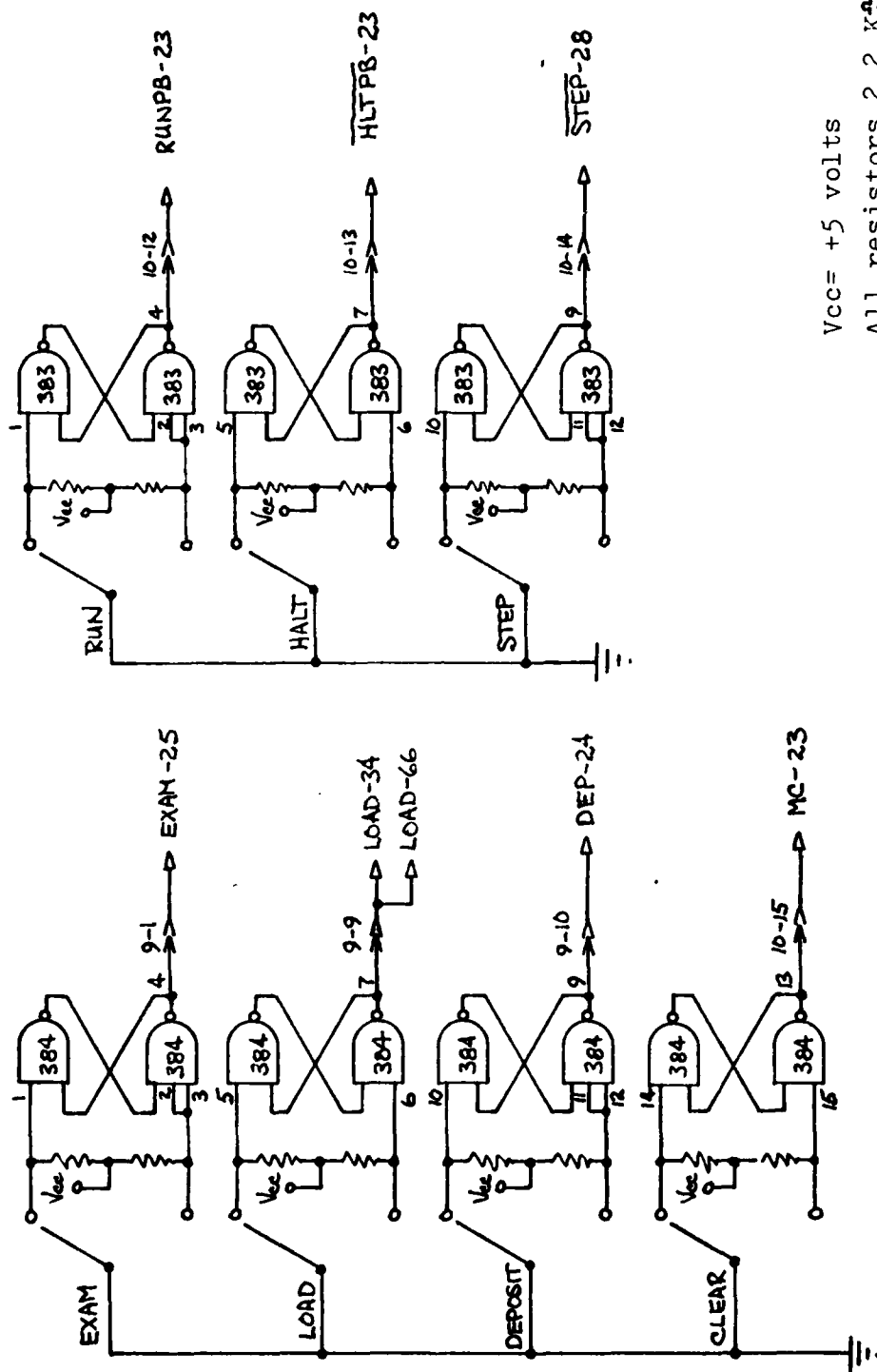
The front panel control switches were required to test all other circuits, so their operation was checked first. Of the seven control switches, four were found to have severe bounce problems. The three remaining control switches were typically used in "preset" or "preclear" circuits and would not indicate bounce even if it were present. All seven of the switches were removed, along with their associated debounce circuitry, and replaced with the switches and debounce circuitry

shown in Figure 26, page 18. A single 74279 dual in-line package (DIP) provided four R-S latches. Thus, two packages were needed to realize all seven debounce circuits. The resistors necessary in the circuits were mounted directly on each switch, and the integrated circuits were mounted on vector board on the rear of the front panel. The debounce circuits were tested after installation and found to operate satisfactorily.

#### Control State Generator

Operation of the single-step control switch allowed the operation of the control state generator to be checked. When the state generator was clocked at speeds above single-step, its timing diagram was correct (see Figure 22, page 19). However, when the state generator was "single-stepped" through its eight states, it was noted that 50% of the time it was in no state at all. Not only did this conflict with timing specifications, but required 16 switch actions to step the state generator through only 8 states.

The state generator was redesigned using a four-bit binary counter (7493), and a 4-line to 16-line decoder (74154) (see Figure 28, page 20). The binary counter is incremented through a natural binary count by the system clock, while the decoder allows 1 of 16 outputs to be selected based on the binary count. Since each output of the decoder is of equal duty cycle, several



Vcc = +5 volts  
All resistors 2.2 K $\Omega$   $\frac{1}{4}$ W.

Fig. 26. Control Switch Debounce Circuits

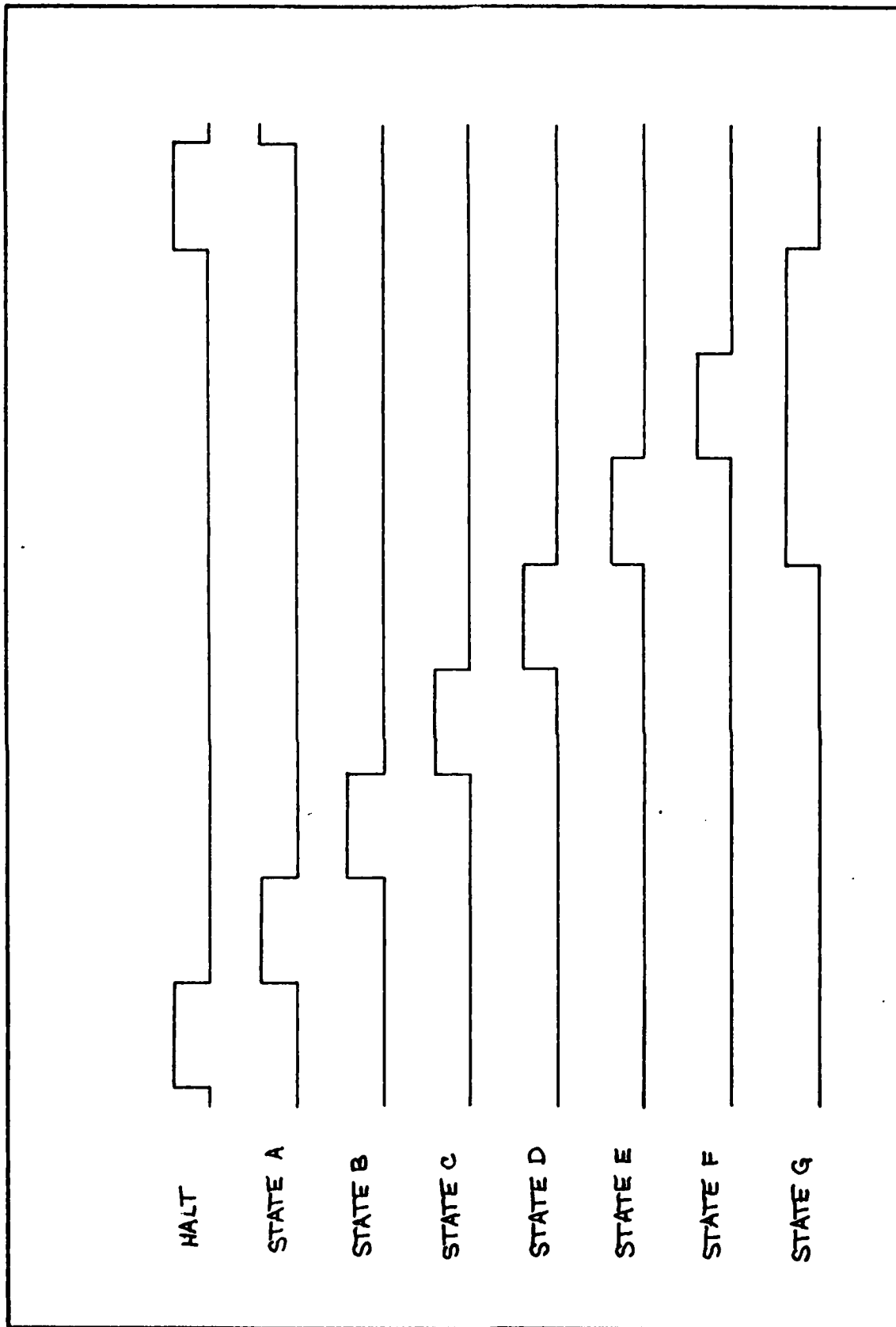


Fig. 22. Control State Generator Timing Diagram

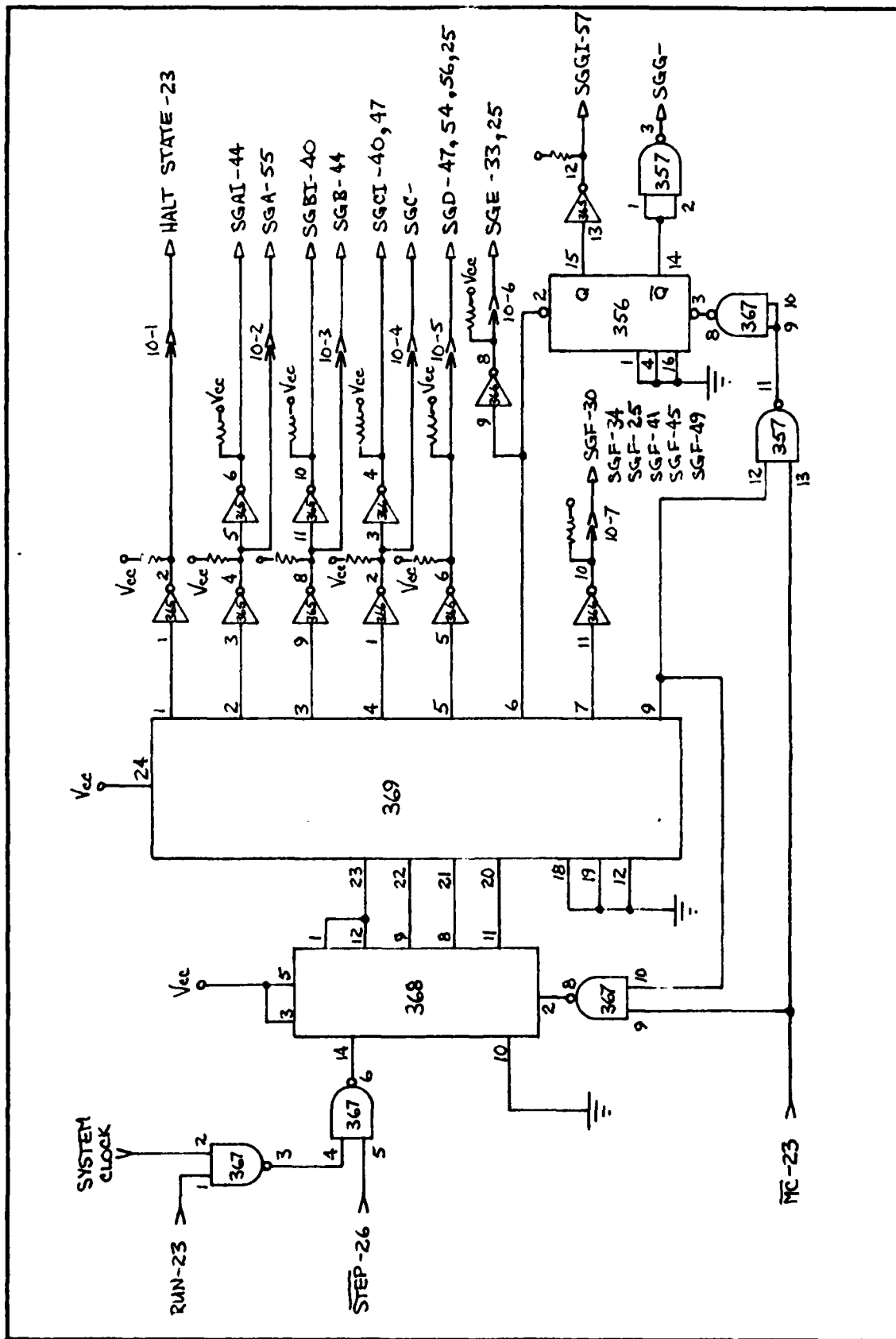


Fig. 28. Control State Generator

outputs are combined to create longer states where necessary. Only 8 of 16 outputs of the decoder are required for the 8 states of the control state generator, so output number 9 is used to reset the binary counter to zero (the Halt state).

In redesigning the state generator to meet specifications, the number of IC's was reduced by about half. Since an MSI package was used in the redesign, it was necessary to relocate the state generator from Module 7 to a module that had 24-pin DIP sockets. Module 16 now houses the state generator. A satisfactory check of the state generator permitted the 256 word scratchpad memory to be checked next.

#### Memory (256 word, scratchpad)

The original design of the Pedagog included a DEC 256 word scratchpad memory (M7319) (see Figure 40, Appendix C). This MOS random access memory was found to be too small to store both the main program code and microcode so a larger, non-volatile 4096 word memory was purchased. The larger memory, however, requires a bus interface to be designed and built before it can be installed. Since the scratchpad memory was plug-compatible with the Unibus, its successful operation would allow the majority of the Pedagog system to be checked out without the bus interface and larger memory.

When microcode was entered into the scratchpad memory, it could not be successfully examined using the front panel switches. Only the first word of memory could be successfully examined.

An analysis of the "examine" and "memory control" circuits revealed a design flaw that caused memory to be read while an address was being loaded. The examine circuit was modified and appears in Figure 25, page 23. The circuit was tested by examining microcode entered in specified locations throughout the scratchpad memory. The circuit successfully examined all locations tested.

#### Run/Halt Flip-Flop

The run/halt flip-flop serves three important functions: 1) it enables the system clock to the state generator when a run condition is present, 2) it disables the system clock to the state generator when a software-controlled Halt command is present, and 3) it drives the run/halt indicators on the front panel. Since the run/halt flip-flop is closely related to the state generator, the redesign of the state generator required that the run/halt flip-flop be redesigned also. Several of the signals used in the old run/halt circuit were no longer needed, so a simpler run/halt flip-flop circuit was designed and implemented (see Figure 23, page 24).

One-half of a dual J-K flip-flop with preset and preclear (7476) was used as the heart of the new circuit. Only the preset and preclear lines of the flip-flop were used while the clock and J-K input lines were tied to ground. A dual retriggerable monostable multivibrator (74123) was strapped to behave as two individual non-retriggerable one-shots, and used to provide

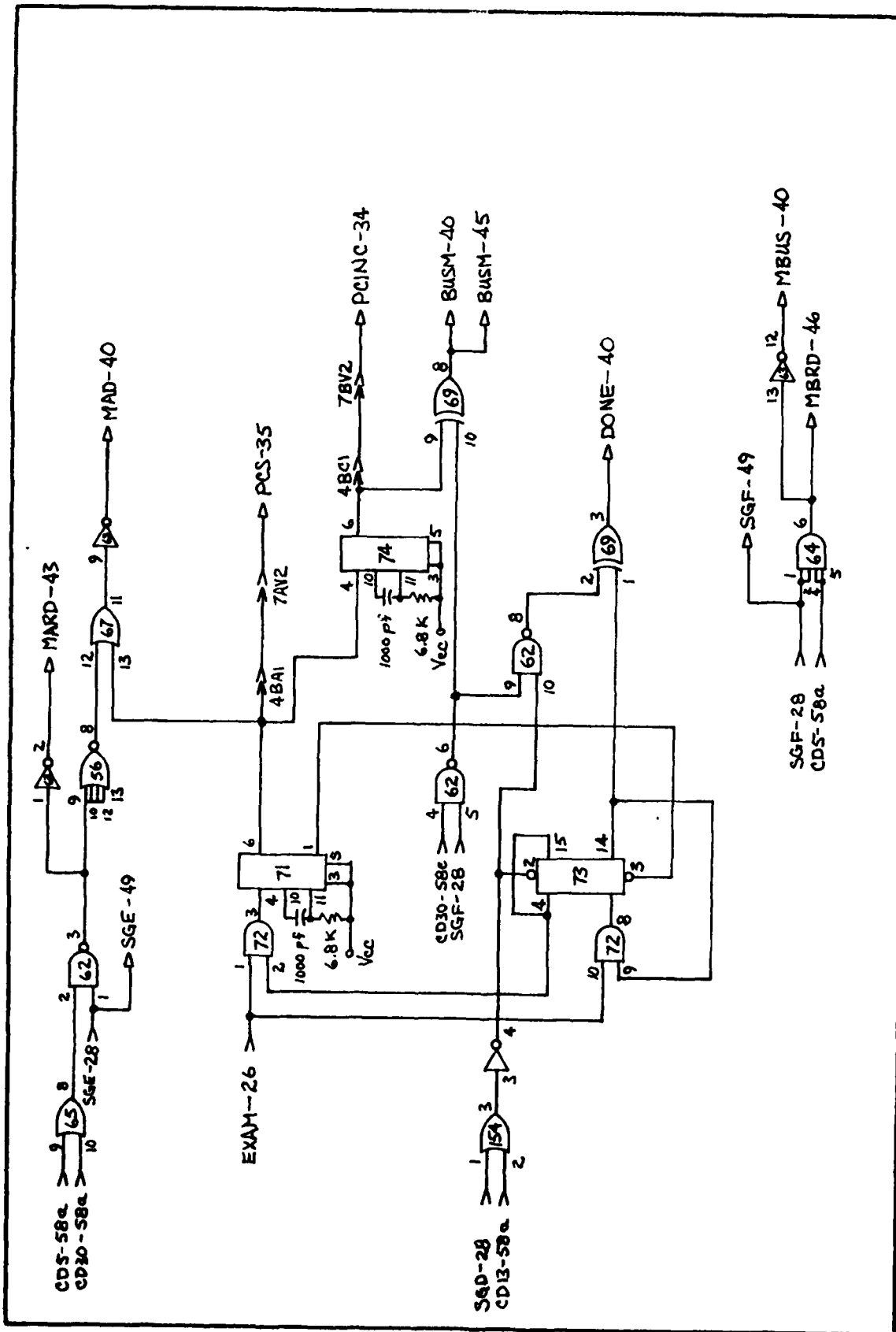


Fig. 25. Examine and Memory Control Circuit

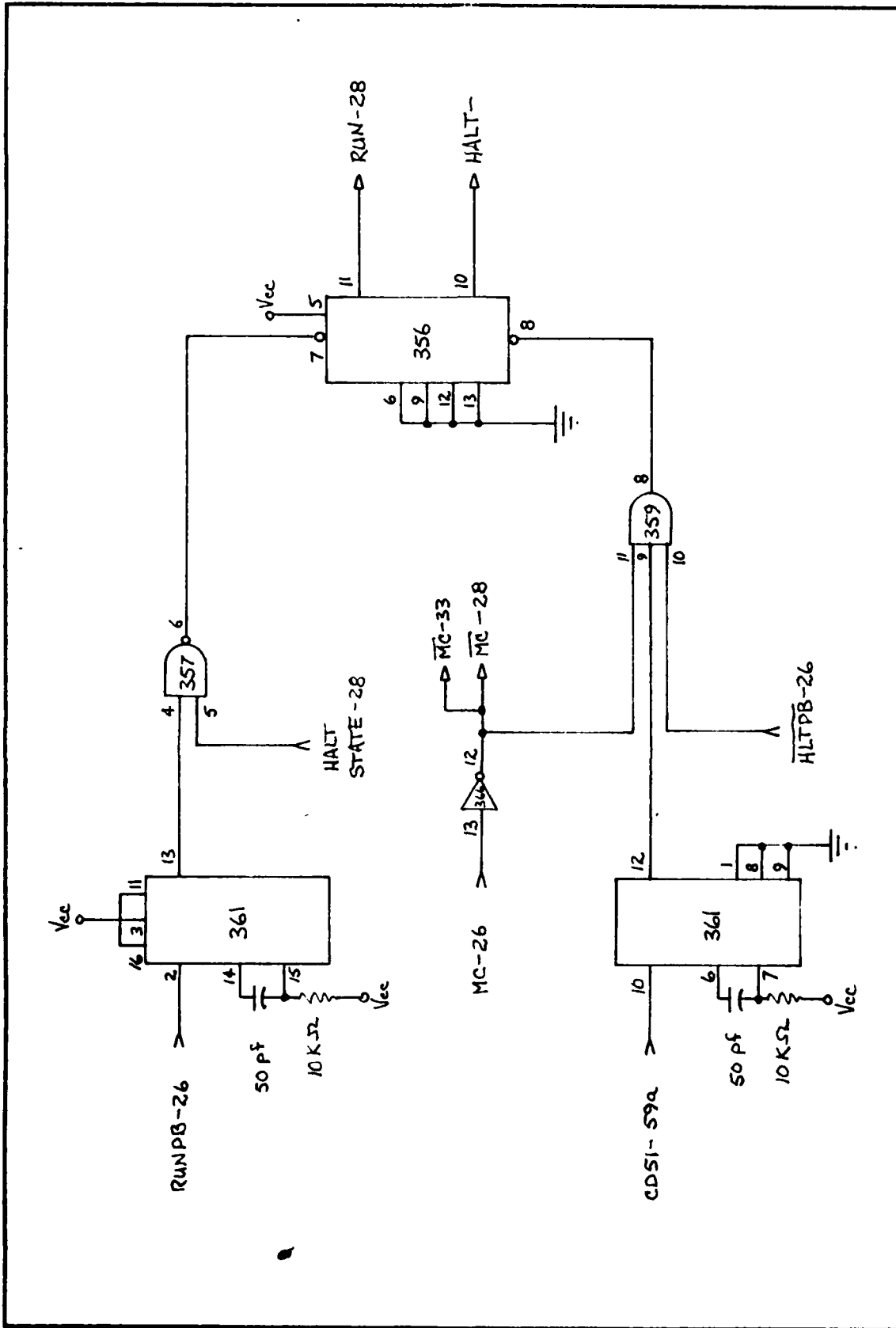


Fig. 23. Run/Halt Flip-Flop and Master Clear Circuit

either a run pulse or a halt pulse to the run/halt flip-flop. As can be seen from the circuit diagram, the machine must be in the Halt state before the "run" switch has any effect. Also, note that either the Master Clear (MC), halt switch (HALT), or Halt instruction (via CD51) can halt the machine. The new circuit was verified by running a short program with the Halt instruction (see Appendix F). The machine successfully halted under software control, and using the HALT and Master Clear switches.

#### Accumulator

After repair of the run/halt flip-flop, the Pedagog was in a condition that would allow short programs to be run. Memory could be accessed and the machine would halt when given a Halt instruction. It was noted, however, that during single-step execution of programs the accumulator display on the front panel was inactive. Several attempts were made to access the accumulator under program control with no success. The accumulator logic diagram was analyzed and it was discovered that during a previous project a change to the Master Clear circuit of the computer had been made but had not been implemented properly in the accumulator circuit. The "clear" lines of the accumulator register (74194) were "active low" and, therefore, required a logic 1 to be cleared instead of a logic 0. Consequently, the accumulator was always being cleared by the Master Clear line.

either a run pulse or a halt pulse to the run/halt flip-flop. As can be seen from the circuit diagram, the machine must be in the Halt state before the "run" switch has any effect. Also, note that either the Master Clear (MC), halt switch (HALT), or Halt instruction (via CD51) can halt the machine. The new circuit was verified by running a short program with the Halt instruction (see Appendix F). The machine successfully halted under software control, and using the HALT and Master Clear switches.

#### Accumulator

After repair of the run/halt flip-flop, the Pedagog was in a condition that would allow short programs to be run. Memory could be accessed and the machine would halt when given a Halt instruction. It was noted, however, that during single-step execution of programs the accumulator display on the front panel was inactive. Several attempts were made to access the accumulator under program control with no success. The accumulator logic diagram was analyzed and it was discovered that during a previous project a change to the Master Clear circuit of the computer had been made but had not been implemented properly in the accumulator circuit. The "clear" lines of the accumulator register (74194) were "active low" and, therefore, required a logic 1 to be cleared instead of a logic 0. Consequently, the accumulator was always being cleared by the Master Clear line.

The addition of an inverter in the line before IC 141 (see Figure 30, page 27) restored the accumulator to proper working order. Under program control the accumulator seemed to operate properly, so testing proceeded to the Instruction Register Decoder circuits.

#### Instruction Register Decoder

After all Pedagog systems seemed to be operating correctly, each main program instruction was executed individually to ensure proper operation. First, all microcode was entered into memory and each instruction was executed using the single-step to check proper microdecoding into memory. It was discovered that two instructions, NOP (no operation) and IAC (increment accumulator), (see Appendix F) decoded to the same microcode in memory.

The problem was traced to an improper IC in the IR Decoder (IRD). A quad AND gate IC (7400) had been installed where a quad OR gate (7432) should have been. The AND gate was replaced and the instructions decoded to their proper place in memory. All other instructions were checked and no problems were found.

#### Microprogramming

The microprogramming to emulate the PDP-8 provides only the 36 basic instructions (see Appendix F). Pedagog uses memory-mapped microprogramming (Ref. 1:30-37) and does not have the ability to execute two instructions as one, as do the later versions of the PDP-8. When the microcode for Pedagog was

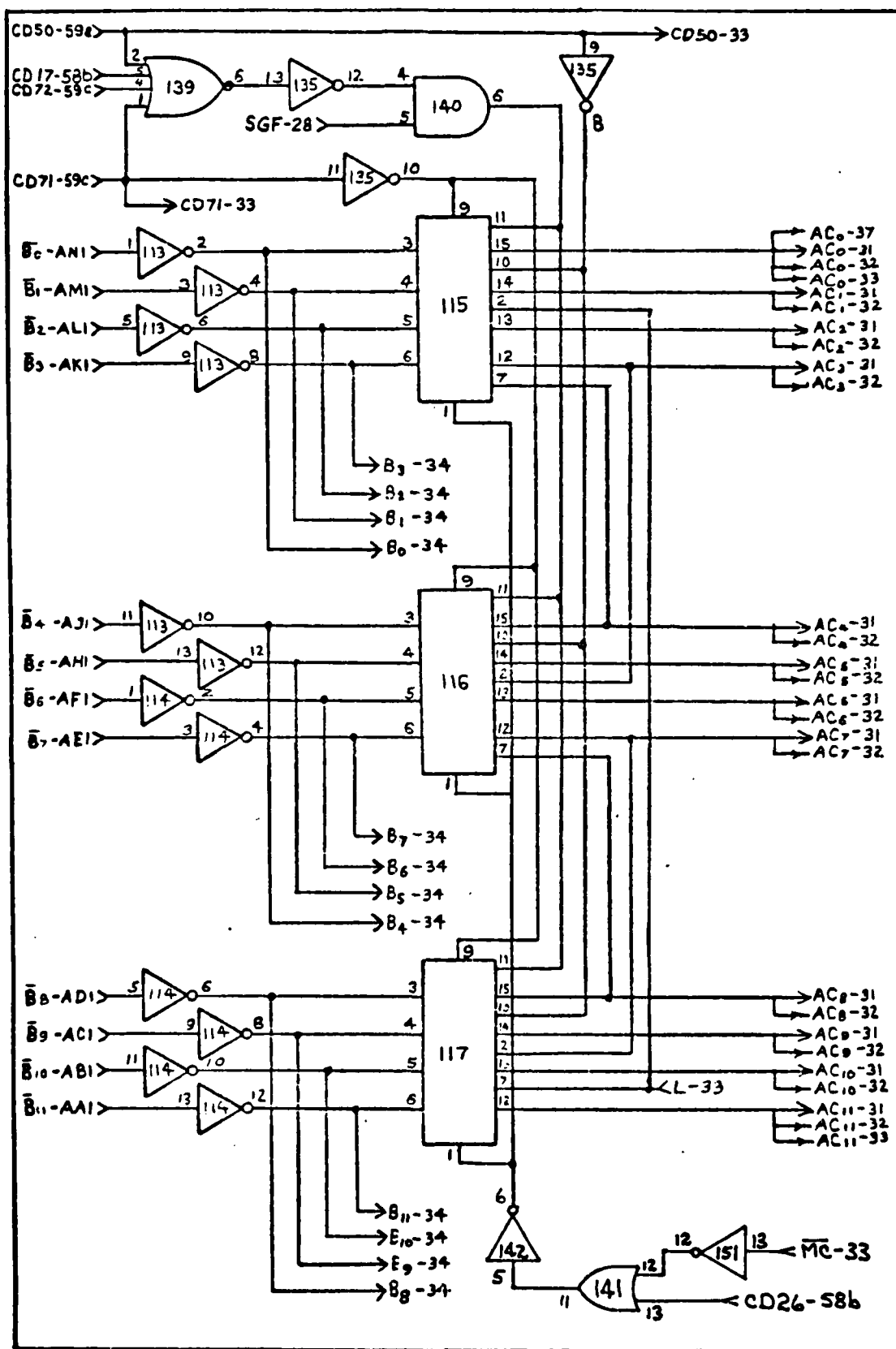


Fig. 30. Accumulator

originally written, it included the instruction "Decrement and Skip if Zero" (DSZ). This instruction is not PDP-8 compatible and was reprogrammed to be "Increment and Skip if Zero" (ISZ). ISZ is PDP-8 compatible and typically requires that the two's complement of a given number be taken before the instruction is used (Ref. 9).

The microcode that was written for Pedagog did contain several errors (Ref. 1:216-221). Rather than list the specific errors, a general warning is given regarding microprogramming. A microword is four octal digits long (12 bits). Pedagog is presently configured so that the first two octal digits of the microword must be  $37_8$  or less, and the last two octal digits must be greater than  $40_8$  and less than  $77_8$ . If, for instance, a number larger than  $37_8$  is used for the first two digits in the microword, the control signals will be erroneous, even though the number may be an acceptable code in the last two digits. The only exception to this is the "no operation" instruction (77). It may be used in either position. The microcode previously written for Pedagog had several errors of this nature. They were corrected and tested and the complete list of microcode to emulate the PDP-8 is given in Appendix K.

#### General Problems

Several small problems were found in Pedagog. Two particular problems that deserve mention were fanout and wire routing.

The fanout for TTL logic is typically 10 loads. In four places in Pedagog the fanout was exceeded by two loads. They were the MAR/MMAR Output Control Switch, MBR/MMBR Output Control Switch, MPC Bus Control Switch, and the PC Output Control Switch (see Appendix C). In the first three cases the signals with fanout problems were from the state generator. When the new state generator was built, buffers were installed in these lines to alleviate the problem. The fourth case, the PC Output Control Switch, operated properly in its present configuration but was buffered to prevent future problems.

The second problem encountered in Pedagog was the wire-wrap technique used. In making connections from pin-to-pin, the wires were often pulled tight around the corner wire-wrap pins. As the wire aged, the insulation evidently chaffed and allowed several short-circuits to occur. The problem was manifested by intermittent flickering on several digits of the octal display while a particular register was selected. It was particularly noticeable when the bus was selected. Often the flickering could be eliminated by flexing Module 7. The fix tended to be only temporary, however, and the flickering usually returned. Rewiring the Pedagog modules was considered, but discarded as impractical. A more practical solution was to bend each of the suspected corner pins inward until the wires were no longer stretched tight around them. No further shorts have been noticed since this action was taken.

### Summary

Several major problems were discovered in Pedagog. The state generator and run/halt flip-flop were found faulty and redesigned. The debounce circuits for the front panel control switches were not reliable and had to be replaced. The Master Clear line to the accumulator was improperly implemented, but was corrected by adding an inverter. The microprogramming that allows the Pedagog to emulate the PDP-8 was tested and verified. Several extensive programs requiring memory reference, register reference, and arithmetic manipulation were written and executed indicating Pedagog II now operates as designed (Ref. 1).

The next chapter of the report describes the redesign of the Pedagog control panel and cabinet.

#### IV. Control Panel and Cabinet

Pedagog II was designed to be a teaching aid. This was a primary factor in the panel layout and cabinet design. All aspects of the machine have been engineered with the student/user in mind. This chapter deals with the design and construction of the Pedagog cabinet, power supplies, and panel. The cabinet is discussed in three parts. They are cabinet layout, on/off switch circuitry, and cooling system. Discussion of the power supplies includes a description of the voltage converter circuit and circuit breaker types and locations. The panel consists of a section on the cabinet top, and a section in the backplane. It will be discussed in two sections: 1) panel layout, and 2) panel backplane.

##### Cabinet Layout

The cabinet of Pedagog was made from a surplus teletype controller. The card rack from the teletype controller was removed and a 19-inch computer slide-mount was installed in its place. The Pedagog backplane was then bolted to the slide-mount. The backplane can be pulled forward to allow easy access to Pedagog circuit cards, or pushed back to allow access to the pins on the backplane. When the slide-mount is pushed in, metal panels snap over the front and rear of the cabinet. The table top is hinged at the rear and can be raised to allow the card rack to be accessed from the top. A table top support brace is

located on the left side and can be locked to hold the table top in an upright position.

Cooling System. Five cooling fans are located in the bottom of the cabinet. Four of the fans cool the Pedagog circuit cards vertically, and one fan is mounted horizontally in the bottom of the cabinet to cool the power supplies. Holes were cut in the solid metal bottom of the slide-mount to allow cooling air to reach the circuit cards. The cooling fans are powered by the 115 VAC line and are controlled by the on/off switch on top of the cabinet. The fans are covered by wire mesh to prevent foreign objects from falling through. Small objects that fall through the mesh can be removed through the snap-out metal panel on the lower front part of the cabinet. A heat sensor bar is mounted directly to the Pedagog card cage. It lies directly above the circuit cards and is wired to an OVERTEMP light mounted with the on/off switch. If an overtemp condition is sensed, the machine will shut down. All circuit breakers and fuses should be checked, and the reason for the overtemp should be determined before turning the machine on. The heat sensor bar can be removed, if necessary, by disconnecting the sensors at the spade lugs. However, the machine will not operate with the heat sensor bar disconnected.

Power Supplies. The Pedagog cabinet provides five power supplies. Four of the supplies are user accessible and the fifth

provides the 28 VDC to operate the on/off switch and its relay. The four useable power supplies are: 1) -5 VDC, 2) +5 VDC, 3) -18 VDC, and 4) +18 VDC. The -5 volt supply is not required by Pedagog and is not used.

The +5 volt supply is the major supply of the system. The +5 volt supply can provide 100 amperes of current and is distributed throughout the Pedagog by a bus bar along the inside rear of the cabinet. A ground bus bar runs parallel to the five volt bus bar. The two bars are only an inch apart and care should be taken that they are not inadvertently shorted while the system is on.

The Pedagog requires a  $\pm 15$  volt power supply for the (proposed) Omnibus. A voltage converter circuit was designed to convert  $\pm 18$  volts to  $\pm 15$  volts. The circuit is shown in Figure 20, page 34. The circuit consists of two voltage regulators (7815 and 7915) which are used to drop the -18 and +18 volt supplies to approximately -15 and +15 volts, respectively. Only the scratchpad memory requires the -15 volt supply, so the current demand is less than one ampere. The +15 volt supply is not used, but will be needed when the bus interface is installed. The voltage converter circuit is rated at 1.5 amperes and is mounted on a phenolic circuit card between the two (existing) power supplies.

Fuses and Circuit Breakers. There are four circuit breakers and three fuses in Pedagog. Three of the four circuit breakers

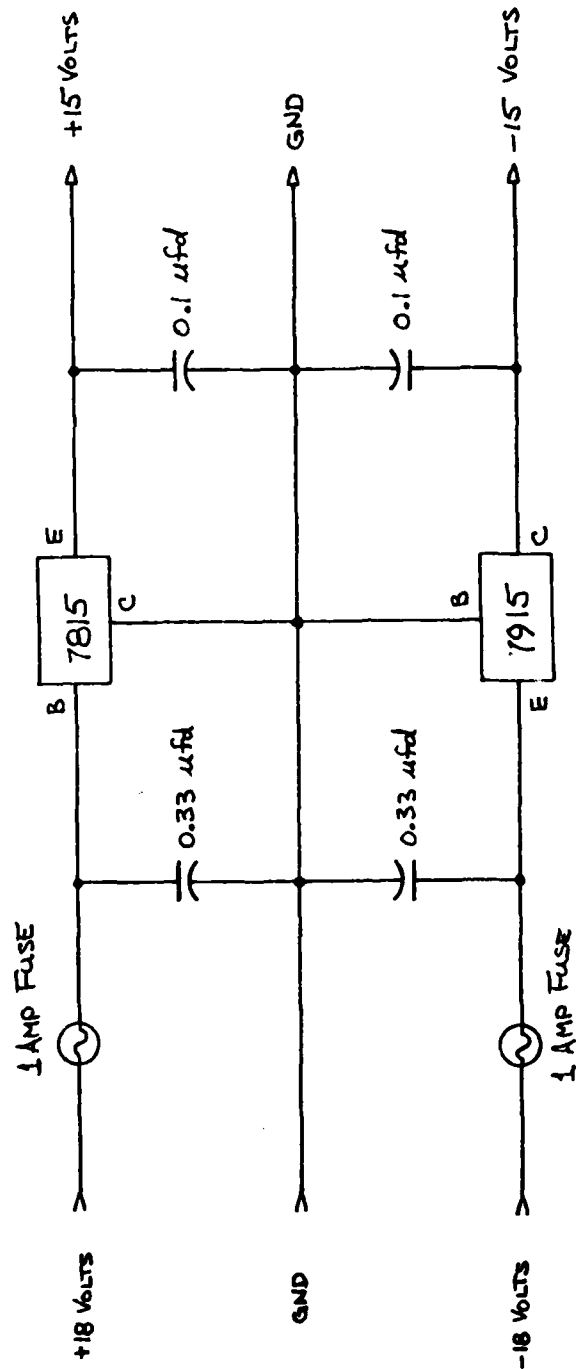


Fig. 20. Voltage Converter Circuit

control alternating current to the machine and are located on the lower left rear (facing the rear) of the cabinet. Three of the circuit breakers are push types and are labeled AUX, 50, and 150 (amperes). The circuit breaker labeled AUX is not used.

The 50 amp circuit breaker controls alternating current on the secondary side of the  $\pm 18$  and -5 volt power supplies. The 150 ampere circuit breaker controls alternating current on the secondary side of the +5 volt power supply. The fourth circuit breaker is a lever-action type and is "on" in the up position. This circuit breaker controls alternating current on the 115 VAC primary side of the power supply transformers.

Three fuses control the direct current in Pedagog. Current in the +5 VDC power supply is controlled by a 15 ampere fuse. It is located on the left side of the backplane and is accessible only from the rear. Each 15 volt supply has a 1 ampere fuse located on the upper part of the voltage converter circuit card. The fuse holders are the snap-in type and are most easily accessed from above.

#### Front Panel

The control panel module is in two sections. The front panel section contains all LED limiting resistors, the control switch debounce circuits, the octal display inverters, and all the hardware associated with the panel. The backplane section (designed in a previous effort) contains the signal synthesizer,

synthesizer multiplexer (MUX), display drivers, and display encoder. The front panel and its design layout will be discussed first (see Figure 18, page 37).

Panel Layout. The front panel has four 12-bit binary displays and one 4-digit octal display. The four binary displays are each dedicated to a particular source. Three of the displays are for the accumulator, program counter, and instruction register. The fourth binary display is dedicated to the Unibus so that each register transfer requiring the Unibus can be observed. The LED displays are in groups of three bits for easy visual conversion to octal. Each LED is connected to the panel by snap-on leads to allow easy replacement.

The octal display is multiplexed for display of eight sources. They are the bus (Unibus), MPC, MMAR, MMBR, MIR, MBR, MAR, and IR. A selector indicator beneath the octal display indicates which source is being displayed. A lamp test switch in the lower left corner of the panel allows all segments of the octal display to be tested. The switch is spring-loaded to the center position and should be pressed down to test the octal display. There is no up position on the switch.

Two rotary switches are located on the front panel. The left switch is an eight-position switch used to select the source of the octal display. The right switch can be used to select any one of four clock rates for the system. The rates are approximately 100 hz, 1 Khz, 10 Khz, and 100 Khz. The lower two

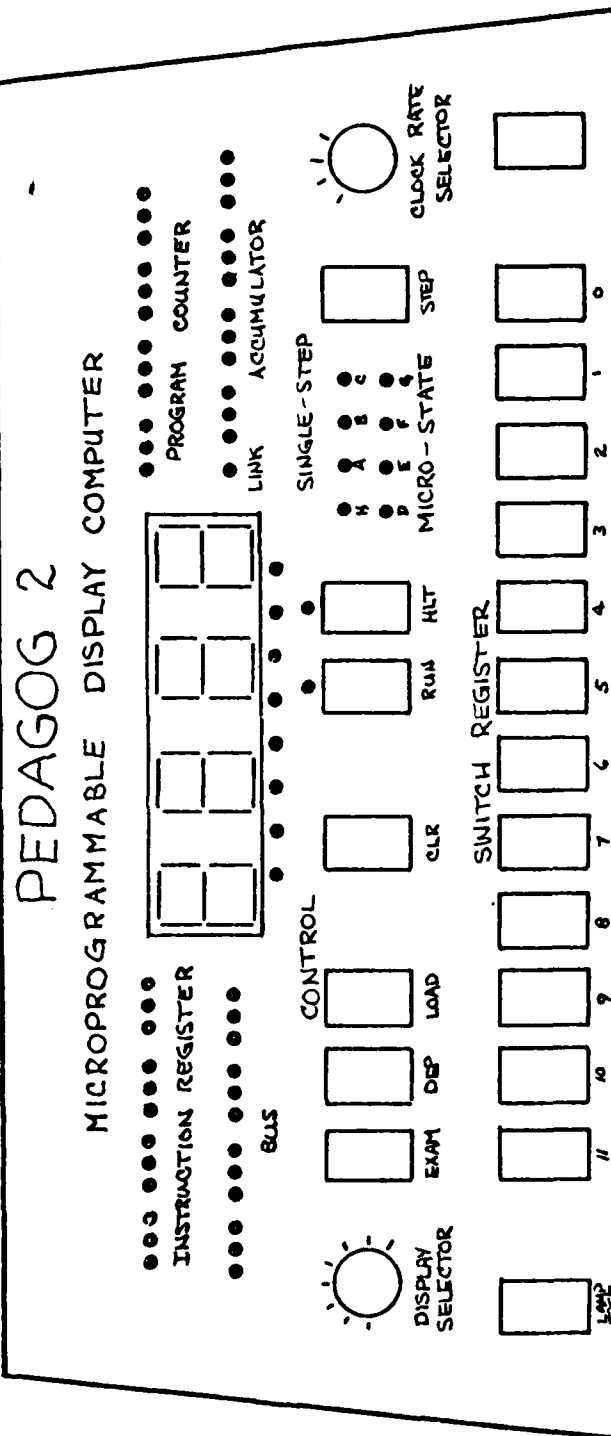


Fig. 18. Control Panel

clock rates are typically used for system debugging. The highest clock rate (100 KHz) should be used for normal operation.

There are two banks of switches on the Pedagog panel. The lower bank is the input to the switch register. The switches are bezel-mounted to the plexiglas face panel and are grouped by three's for easy octal encoding. The "up" position is a logic 1 and the "down" position is a logic 0. The upper bank of switches are also bezel-mounted and control the various Pedagog functions. They are DEPOSIT, EXAMINE, LOAD, RUN, HALT, CLEAR, and STEP. Each of these functions is explained in detail in Chapter 2.

LED current limiting resistors are in "dip" packs on the rear of the panel. Resistor values of 470 ohms were used for all binary displays, and 270 ohm resistors were used for all indicators (see Appendix C for wiring schematics). Two hex inverter IC's (7404) are located on the rear panel and serve to invert the signals presented to the octal display. Debounce circuits for the control switches are also on the rear panel and operate as indicated in Chapter 3.

Ten ribbon cables carry control and data information from the backplane to the control panel. The cables run through a slot in the bottom of the panel housing and down to the panel circuit cards that interface to the computer. These cables are of ample length to allow the backplane to slide out or the table top to open without disconnecting the cables. The cables are

numbered and the control signals for each cable are listed in Appendix G.

Panel Backplane Section. The panel backplane section includes the signal synthesizer, multiplexer, multiplexer encoder, and inverter/driver (see Figure 19, page 40 ). The panel backplane section is located on two half modules, 12 and 13. The ten connecting cables from the panel provide control and data information to the Unibus through this interface.

The signal synthesizer duplicates certain registers (MBR, MMAR, MMBR, IR, MIR) that extract their information from the Unibus. These registers are actually copies of the original registers in the Pedagog (Ref. 2:8-10). Insufficient pins were available on the backplane to bring the actual outputs of these registers to the panel. The signal synthesizer duplicates these registers by using the same control signals as the originals to gate the proper information from the bus. Consequently, the contents of the MBR, MMAR, MMBR, IR, and MIR are exactly duplicated in five registers in the signal synthesizer. The signal synthesizer also duplicates the MPC register using the same register inputs as the original. The outputs from the synthesizer are sent to the multiplexer (MUX) where they are selected for display by the front panel rotary switch.

The MUX uses select data from the Control Panel Selector Switch to enable the desired register display. The MUX is situated on modules 12 and 13, each of which handles 6 bits of data

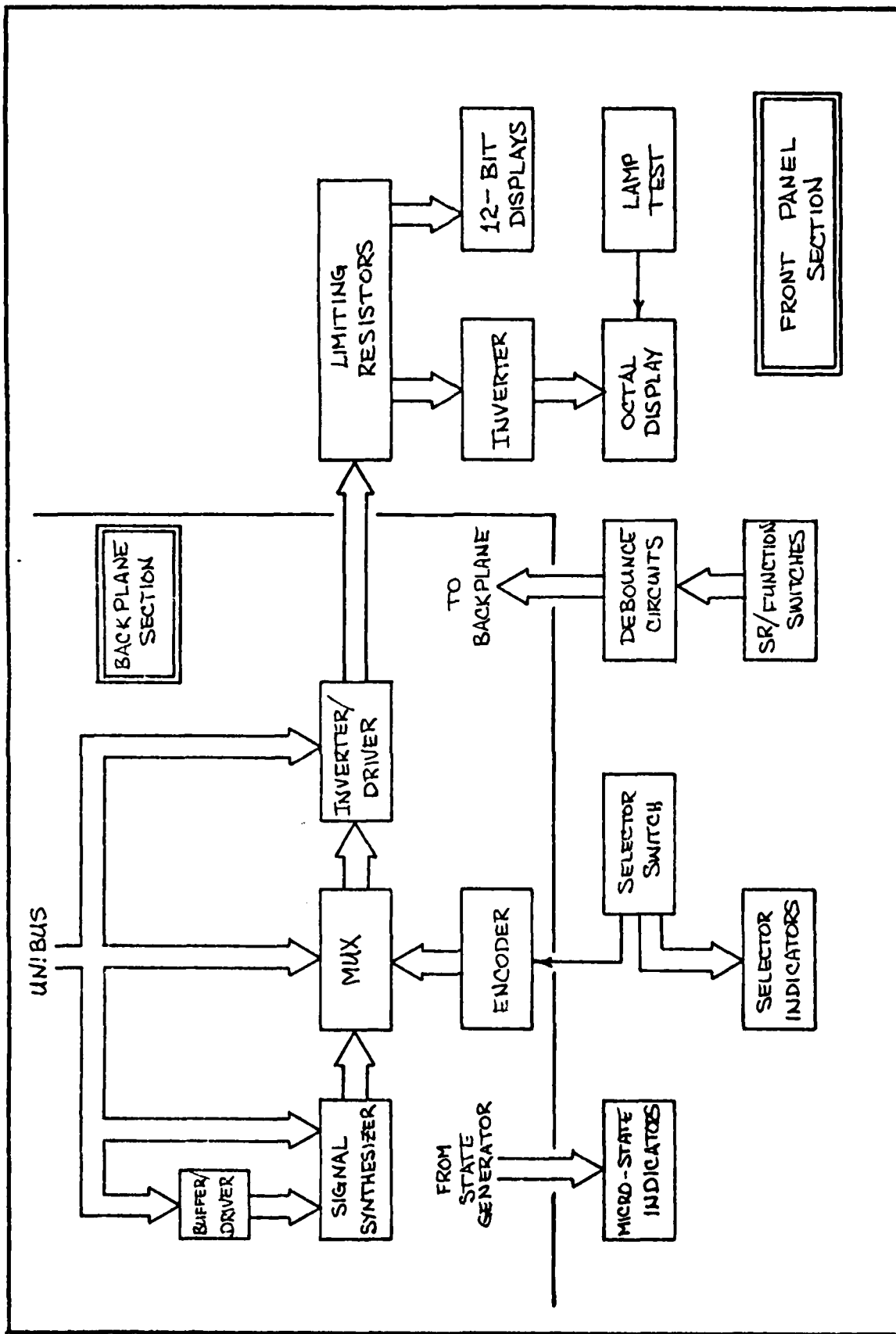


Fig. 19. Control Panel Block Diagram

(0 to 5, 6 to 11) and consists of six 74151 multiplexers per module. Eight signals are input to the MUX. An input from the Control Panel Selector Switch position corresponding to each signal is fed to the input of a 74148, priority encoder, which puts out a 3-bit binary code to gate the corresponding signal out of the MUX to drive the display.

The inverter/driver is also located on Modules 12 and 13. This circuit obtains signals from the Unibus, PC, and AC together with generated signals from the synthesizer and the MUX to drive the display LED's.

#### Summary

This chapter has described the Pedagog cabinet and control panel. The cabinet is a surplus teletype controller with built-in cooling fans. Two power supplies and a voltage converter circuit provide  $\pm 15$  VDC and +5 VDC to Pedagog. Machine circuitry is accessible from the front, rear, and top, allowing ease of maintenance.

Three circuit breakers and three fuses are used to limit current.

The control panel module is in two sections. The panel section is inset in a bakelite housing on the table top and contains the function switches and displays necessary to control Pedagog. The backplane section of the control panel module is located in the cabinet and contains the buffers, drivers, and

other circuitry necessary to operate the panel.

The next chapter concludes this analysis of Pedagog II.  
Several recommendations for continued study are included.

## V. Recommendations and Conclusions

Pedagog II is the first machine designed, realized, and implemented at AFIT using integrated circuit technology. Even though the machine is fully operational for the first time in several years, it has yet to undergo the rigorous tests that are provided during daily use. As with any engineering design, flaws will be discovered which must be eliminated. Great care was taken to insure this machine performs according to the specifications in this and other efforts. The author feels this has been accomplished.

### Recommendations

Unibus/Omnibus Interface. The effort by Parris (Ref. 3) to add a bus interface to Pedagog has not been realized. An attempt to construct the interface was made in this effort. Errors were discovered in the interface design, and, consequently, the bus interface has not been completed. The results of the effort to realize the bus interface are discussed in detail in Appendix H. A summary of those recommendations is given here.

The bus interface design by Parris was only a partial interface. It would allow only a memory to be added and not other PDP-8 modules. The bus interface should be designed as a total bus interface. That is, an interface that would allow the use of the DATA portion of the Omnibus, as well as the MA (memory address) and MD (memory data) portions. This would

allow any Omnibus compatible module to be added to Pedagog. The number of modules added, of course, would be limited by the number of slots available in the Pedagog backplane. With the memory and its associated circuit cards installed, only one additional backplane slot is available. This slot could be used for the I/O Controller module (M8650). Thus, the memory of the Pedagog could be expanded to 4096 words, and the I/O Controller would allow batch loading of the Pedagog microcode and main programs.

Instructions. A provision to expand the instruction set of the machine was incorporated by Beck and Hartman during the original design. There are eleven blocks of memory reserved for added microprograms. The instruction words are predefined that will decode to these eleven blocks of memory. By using microprogramming finesse, these added instructions can greatly expand the capability of Pedagog. For example, multiplication and division routines could be added. The list of possible additions to the instruction set is infinite because the microprograms can be altered freely by the programmer.

Uses. The use of Pedagog II is presently limited by its small memory and limited I/O capability. Until the 4096 word memory is installed, practical use of the machine would be as a demonstrator for microprogramming or a laboratory tool, rather

than as a full general purpose computer. The 256 word memory should be more than sufficient for microprogramming instruction and experimentation.

The lack of a high speed I/O device makes use of the machine cumbersome. All data, main programs, and microcode must presently be entered through the front panel Switch Register. Given the volatile nature of the memory, reloading the machine becomes monotonous. Therefore, short projects requiring one or two sessions at the machine are most practical.

Pedagog would serve as an ideal introduction to the PDP-8. Use of the machine would allow a student to gain familiarity with the PDP-8 instruction set, its register architecture, and its front panel functions. Until the capabilities of Pedagog are expanded, its uses should remain in this area.

### Conclusions

In this effort Pedagog II was made into a working computer system. All of the objectives of this project were accomplished except the expansion of the Pedagog memory through the construction of a bus interface. The large number of flaws that had to be corrected in order to make the machine operational left insufficient time to complete the interface. The addition of an attractive panel complements the utility of the machine's desk top and cabinet. Power supplies that are needed for the addition of the bus interface are installed and ready for use.

Pedagog II has been a difficult, but rewarding challenge. Many pitfalls were encountered and conquered. Many lessons have been learned, and an intimate knowledge of hardware design has been achieved. The fact that Pedagog II is operational after many years of labor can only be truly appreciated by the designers and contributors of the many projects. The knowledge gained by the author through this endeavor will hopefully be multiplied many times by student use of Pedagog in the classroom and laboratory.

### Bibliography

1. Beck, Richard A. and Richard L. Hartman. Design, Realization and Implementation of PEDAGOG II. Unpublished Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, December 1974.
2. McKay, Hugh G. and Peter G. vonGlahn. Design, Realization and Implementation of PEDAGOG II Control Panel. Unpublished EE 6.50 Project Report. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, Undated.
3. Parris, Winfred G. PEDAGOG II Enhancement and Expansion. Unpublished Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, December 1978.
4. Chamberlain, Leroy B. Design and Simulation of a Small General Purpose Digital Computer. Unpublished Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, March 1972.
5. Digital Equipment Corporation. RTM Register Transfer Modules. PDP-16 Logic Module Manual. Maynard, Massachusetts, Digital Equipment Corporation, 1973.
6. Digital Equipment Corporation. PDP8/E, PDP8/F, and PDP8/M Processor Maintenance Manual Volume 1. Maintenance Manual. Maynard, Massachusetts, Digital Equipment Corporation, 1973.
7. Digital Equipment Corporation. PDP8/E Interface Manual (Preliminary). Interface Descriptions. Maynard, Massachusetts, Digital Equipment Corporation, 1970.
8. Logic Products Group. Hardware/Accessories Catalog. Product Catalog. Mailborough, Massachusetts, Digital Equipment Corporation, 1975.
9. Digital Equipment Corporation. Introduction to Programming. Describes PDP8 operations. Maynard, Massachusetts, Digital Equipment Corporation, 1975.

## Appendix A

### Module Number Assignments and Descriptions

### Module Number Assignments and Descriptions

Throughout this thesis, reference is made to pre-assembled DEC RTM's which are used in the design. These are special purpose modules designed by DEC which accomplish a specific task. In addition to these modules, blank modules were used.

The blank module number assignments used to implement Pedagog II are listed below with a brief description of the module type (quad, dual, or single).

<u>Module Number</u>	<u>Assignment</u>	<u>Card Type</u>
1	IR, IR Decoder	dual
2	IR Decoder	dual
3	IR Decoder, MPC	dual
4	Miscellaneous circuits	dual
5	MIR, Field A Decoder	dual
6	Field B Decoder, Increment PC	dual
7	Accumulator, PC	quad
8	MAR, MMAR, MBR, MMBR	quad
9	Arithmetic Logic Unit (M7300)	dual
10	Arithmetic Logic Unit (M7301)	dual
11	256 word, Scratchpad memory	dual
12	Signal syn, MPC syn, Multiplexer	dual
13	Signal syn, MPC syn, Encoder	
	Buffer/Driver, Multiplexer	dual

<u>Module Number</u>	<u>Assignment</u>	<u>Card Type</u>
14	Unibus/Omnibus interface (proposed, see Appendix H)	quad
15	Unibus/Omnibus interface (proposed, see Appendix H)	quad
16	Control State Generator	dual

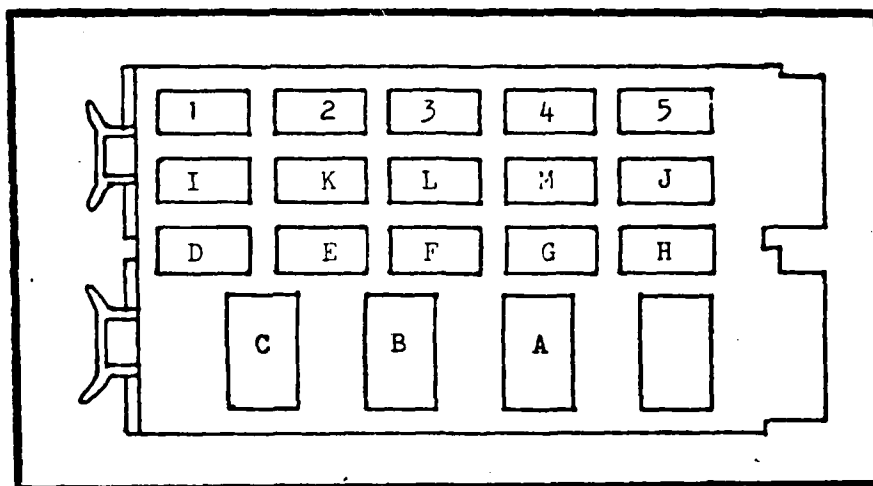
Appendix B

Integrated Circuit Locations and Types

### Integrated Circuit Locations and Types

This appendix describes the locations and types of integrated circuits used in Pedagog II. In the case where no type is listed by a location, no integrated circuit is used. Included in this appendix are the IC locations and types proposed for the Omnibus/Unibus interface by Parris (see Appendix H). These figures are also available in Appendix H for ease of reference.

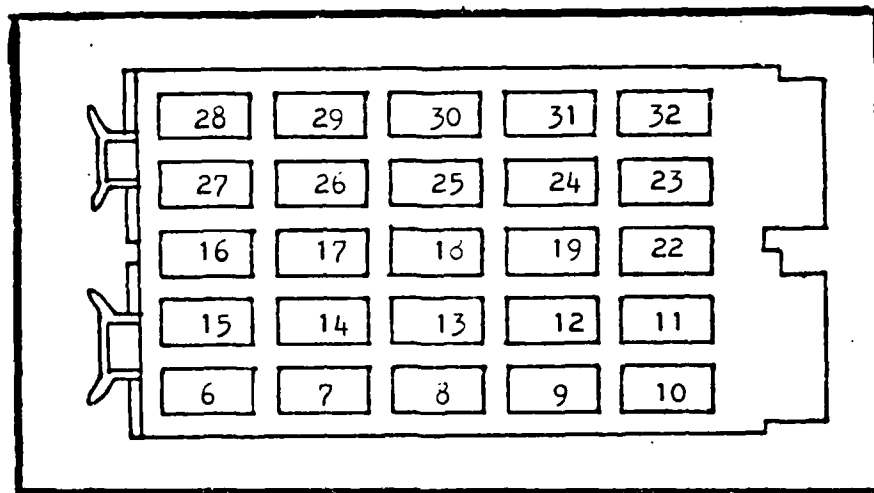
# Instruction Register and Instruction Register Decoder



Module 1

<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>
A	SN74154	J	SN7404
B	SN74154	K	SN7475
C	SN74154	L	SN7475
D	SN7404	M	SN7475
E	SN7404	1	SN7432
F	SN7404	2	SN7432
G	SN7404	3	SN7432
H	SN7404	4	SN7432
I	SN7404	5	SN7432

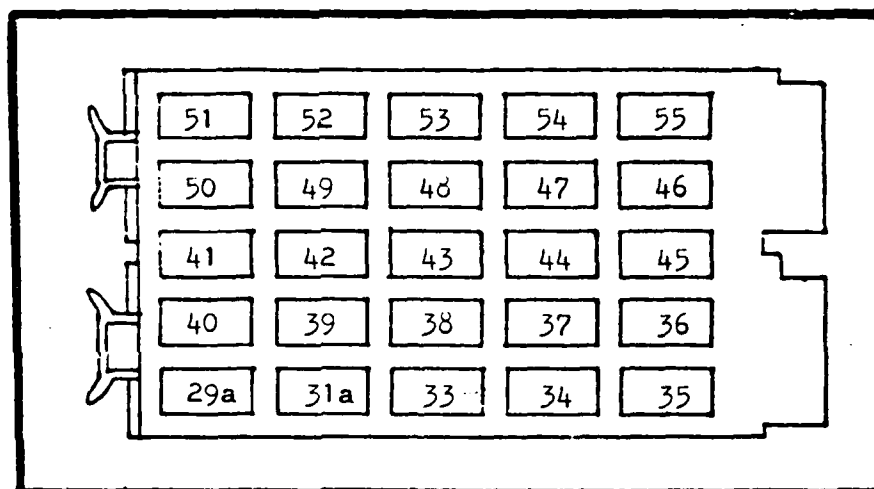
# Instruction Register Decoder



Module 2

<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>
6	SN7432	19	SN7408
7	SN7432	22	SN7432
8	SN7432	23	SN7408
9	SN7432	24	SN7425
10	SN7432	25	SN7408
11	SN7432	26	SN7408
12	SN7432	27	SN7425
13	SN7432	28	SN7432
14	SN7421	29	SN7404
15	SN7408	30	SN7425
16	SN7408	31	SN7432
17	SN7425	32	SN7408
18	SN7404		

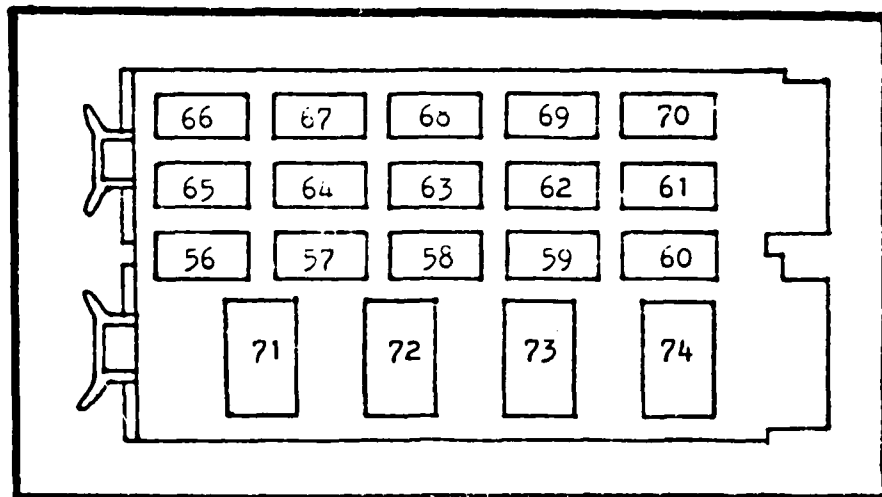
# Instruction Register Decoder and MPC



Module 3

<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>
29a	SN7404	44	SN7432
31a	SN7432	45	SN7404
33	SN7408	46	SN7425
34	SN7408	47	SN7408
35	SN7404	48	SN7408
36	SN7425	49	SN7432
37	SN7432	50	SN74197
38	SN7408	51	SN74197
39	SN7408	52	--
40	SN7425	53	SN7438
41	SN7425	54	SN7438
42	SN7408	55	SN7438
43	SN7421		

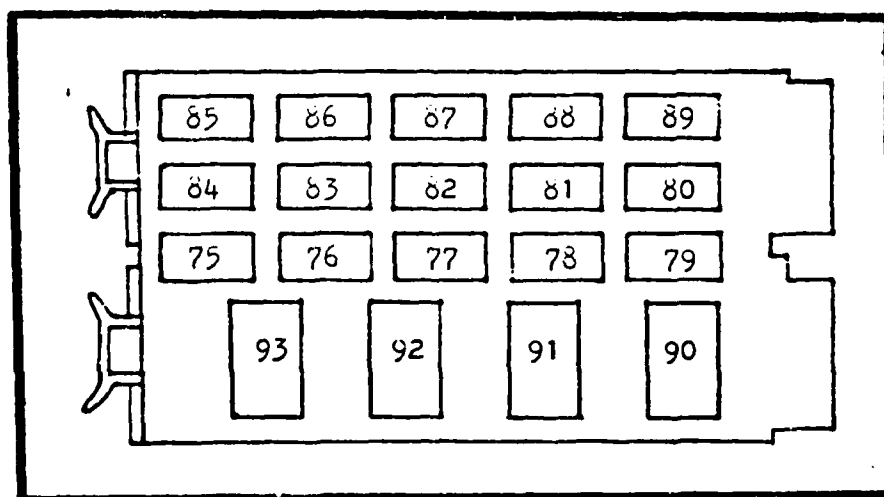
# Miscellaneous Circuits



Module 4

<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>
56	SN7425	66	SN7408
57	SN7406	67	SN7432
58	SN7406	68	SN7400
59	SN7402	69	SN7486
60	SN7402	70	SN74121
61	SN7402	71	SN74121
62	SN7400	72	SN7408
63	SN7404	73	SN7476
64	SN7421	74	SN74121
65	SN7432		

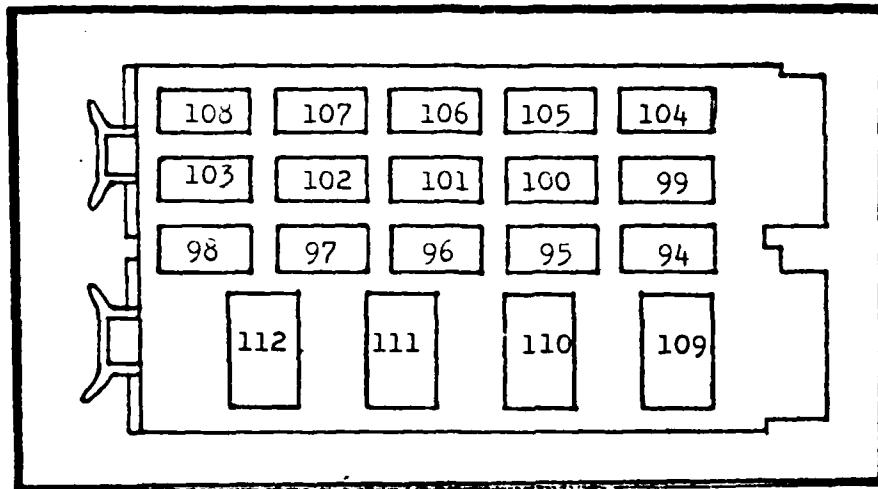
# MIR and Field A Decoder



Module 5

<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>
75	SN7404	85	SN7408
76	SN7475	86	SN7408
77	SN7475	87	SN7408
78	SN7475	88	SN7408
79	SN74155	89	SN7408
80	SN7404	90	SN74154
81	SN7404	91	SN7408
82	SN7408	92	SN7404
83	SN7408	93	SN74121
84	SN7404		

# Field B Decoder and Increment Program Counter

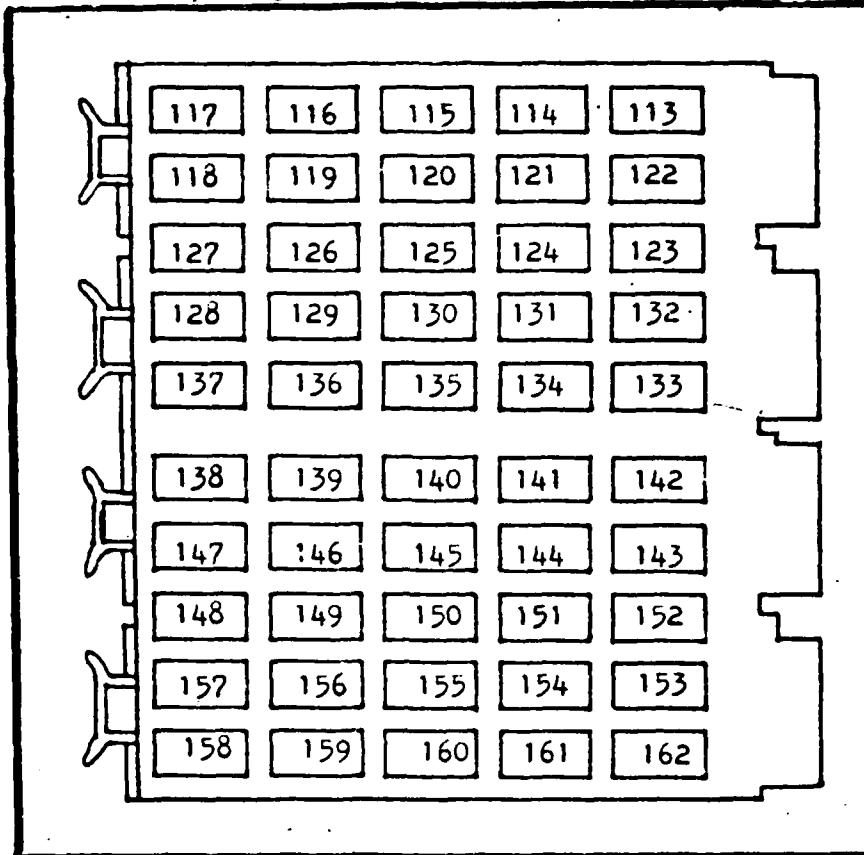


Module 6

<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>
94	SN7404	104	SN7408
95	SN7404	105	SN7404
96	SN7408	106	SN7408
97	SN7408	107	SN7425
98	SN7408	108	SN7408
99	SN7408	109	SN74154
100	SN7408	110	SN74155
101	SN7408	111	SN7432
102	SN7408	112	SN7404
103	SN7408		

# Module 7

Accumulator, Program Counter, Control State Generator

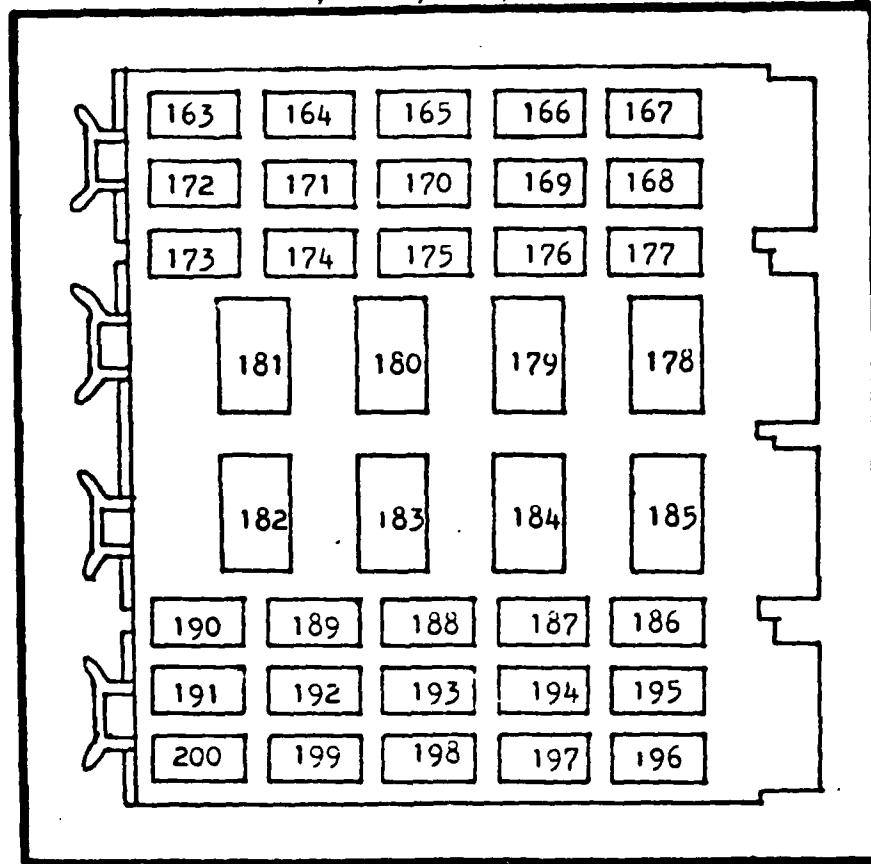


<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>
113	SN7404	123	SN7408
114	SN7404	124	SN74197
115	SN74194	125	SN74197
116	SN74194	126	SN74197
117	SN74194	127	SN7408
118	SN7408	128	SN7408
119	SN7408	129	SN7408
120	SN7408	130	SN7432
121	SN7408	131	SN7432
122	SN7408	132	SN7432

<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>
✓ 133	SN7416	148	--
✓ 134	SN7416	149	SN7408
✓ 135	SN7404	150	--
136	SN7432	2 151	SN7404
137	SN7425	152	--
138	SN7425	153	SN7417
139	SN7425	154	SN7432
140	SN7408	155	SN7432
141	SN7432	156	--
142	SN7404	157	--
143	SN7476	158	--
144	SN7408	159	SN74121
145	SN7425	160	--
146	SN7404	161	SN7400
147	SN7432	162	--

# Module 8

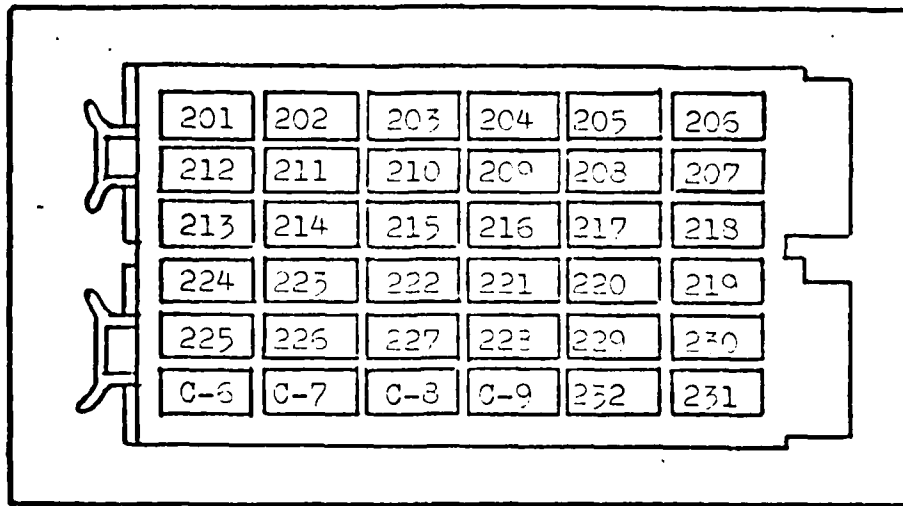
MAR, MMAR, MBR, MMBR



<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>
163	SN7432	173	--
164	SN7432	174	SN7408
165	SN7432	175	SN7408
166	SN7404	176	SN7404
167	SN7432	177	SN7404
168	SN7408	178	SN74116
169	SN7408	179	SN74116
170	SN7406	180	SN74116
171	SN7408	181	SN74116
172	--	182	SN74116

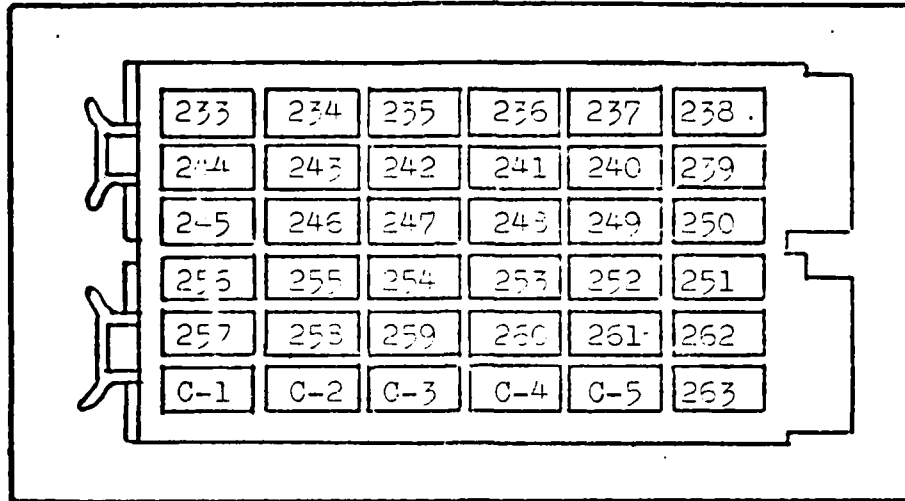
<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>
183	SN74116	192	SN7408
184	SN7416	193	SN7408
185	SN7416	194	SN7408
186	SN7432	195	SN7432
187	SN7408	196	SN7432
188	SN7408	197	SN7432
189	SN7408	198	SN7432
190	SN7404	199	SN7432
191	SN7432	200	SN7408

Module 12. Control Panel Interface Board 1



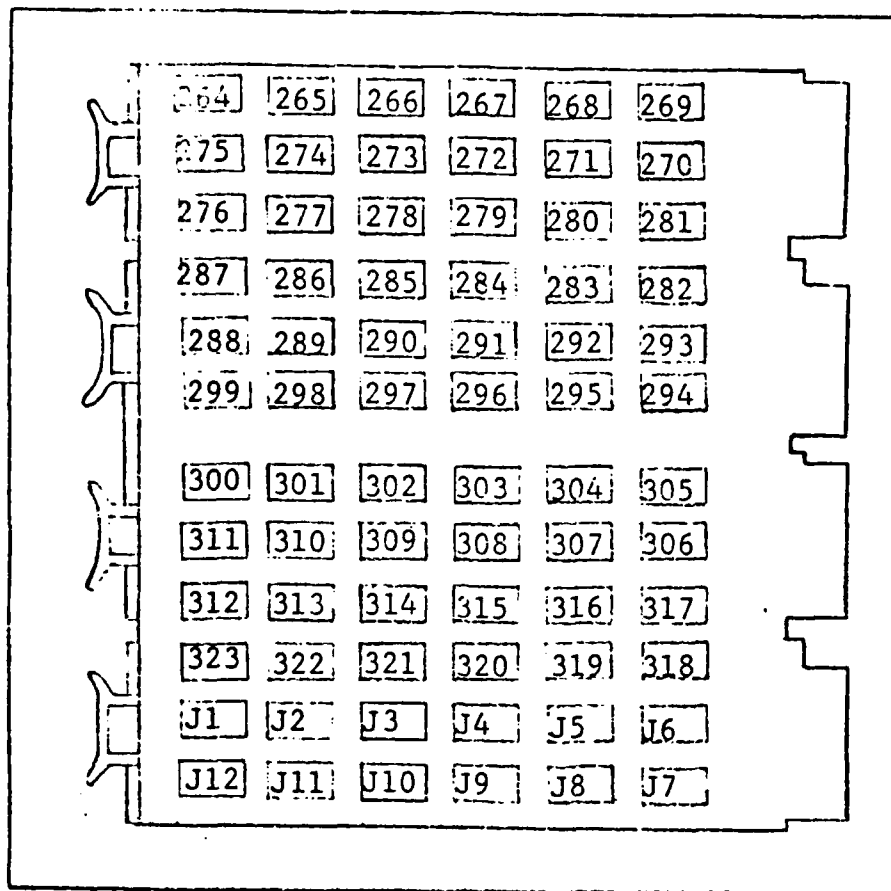
<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>
201	SN74151	212	SN74151	223	SN7416
202	SN74151	213	SN74151	224	SN7416
203	SN7475	214	SN74151	225	SN7416
204	SN7475	215	SN7475	226	SN7416
205	--	216	SN7475	227	--
206	--	217	--	228	SN7416
207	--	218	--	229	SN7416
208	--	219	--	230	--
209	SN7475	220	--	231	SN74197
210	SN7475	221	--	232	SN7416
211	SN74151	222	SN7475		

# Module 13. Control Panel Interface Board 2



<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>
233	SN74151	244	SN74151	256	SN7416
234	SN74151	245	SN74151	257	SN7416
235	SN7475	247	SN7475	258	SN7416
236	SN7475	248	SN7475	259	SN74148
237	--	249	--	260	--
238	--	250	--	261	SN7417
239	--	251	--	262	SN7416
240	--	252	SN74197	263	SN7416
241	SN7475	253	SN7475		
242	SN7475	254	SN7475		
243	SN74151	255	SN7416		
		256			

# Omnibus/Unibus Interface

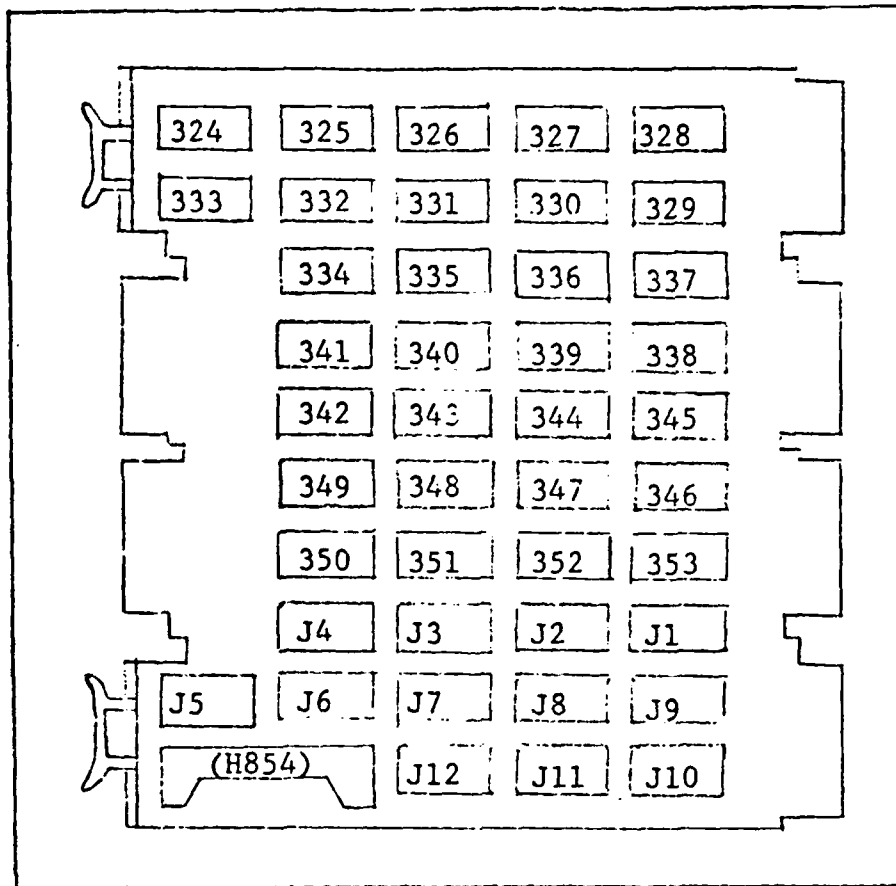


W968 (Module 14) Layout

<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>
264	SN7475	274	SN7475	284	SN7475
265	SN7475	275	SN7475	285	SN7475
266	SN7475	276	SN7475	286	SN7475
267	SN7409	277	SN7409	287	SN7409
268	SN7409	278	SN7409	288	SN7409
269	SN7409	279	SN7409	289	SN7409
270	SN7420	280	SN74121	290	SN7474
271	SN7404	281	SN74121	291	SN74121
272	SN74121	282	SN7401	292	SN7475
273	SN7474	283	SN7408	293	SN7475

<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>
294	SN7475	304	SN7409	314	
295	SN7409	305	SN7409	315	
296	SN7409	306	SN7401	316	
297	SN7409	307	SN7474	317	
298	SN7410	308	SN7404	318	
299	SN7432	309	SN7403	319	
300	SN7475	310	SN7400	320	
301	SN7475	311	SN7401	321	
302	SN7475	312	SN7433	322	
303	SN7409	313		323	

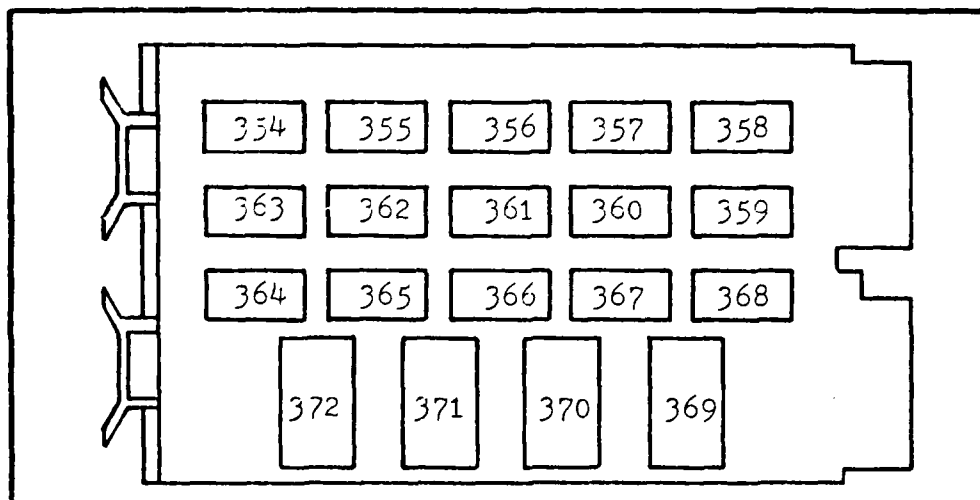
# Omnibus/Unibus Interface



W966 (Module 15) Layout

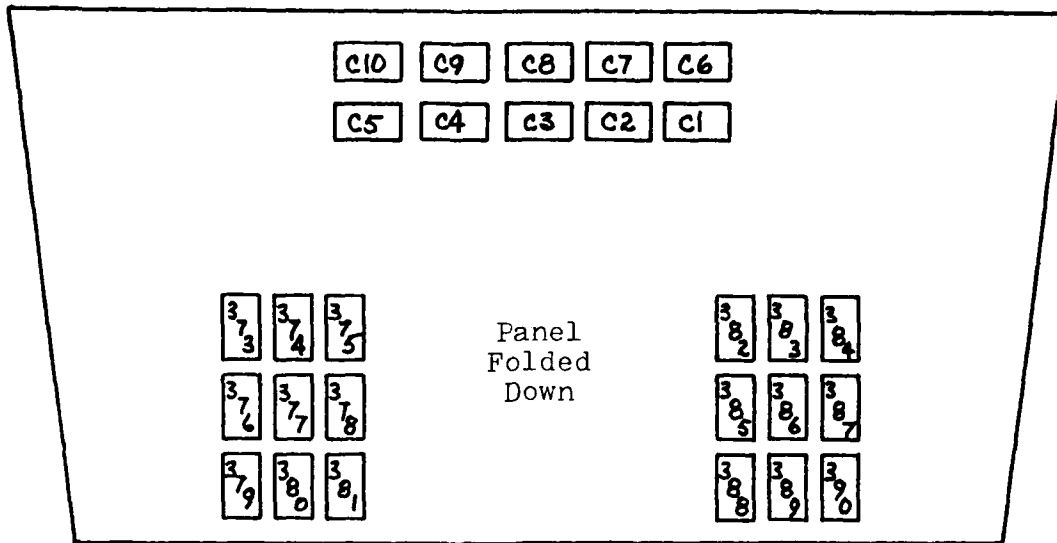
<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>
324		334		344	
325		335		345	
326		336		346	
327		337		347	
328		338		348	
329		339		349	
330		340		350	
331		341		351	
332		342		352	
333		343		353	

Module 16. Control State Generator, Run/Halt Flip-Flop



<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>
354	--	363	--
355	--	364	--
356	SN7476	365	SN7416
357	SN7400	366	SN7416
358	Cable 10	367	SN7400
359	SN5411	368	SN7493
360	Components	369	SN74154
361	SN74123	370	--
362	Components	371	--
		372	--

# Backpanel Module



<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>
373	--	382	--
374	--	383	SN74279
375	--	384	SN74279
376	--	385	270 $\Omega$ DIP
377	SN7404	386	270 $\Omega$ DIP
378	SN7404	387	270 $\Omega$ DIP
379	470 $\Omega$ DIP	388	470 $\Omega$ DIP
380	470 $\Omega$ DIP	389	470 $\Omega$ DIP
381	470 $\Omega$ DIP	390	470 $\Omega$ DIP

Appendix C

Logic Diagrams

## Logic Diagrams

This appendix contains the logic diagrams for the entire Pedagog system. The signals have been labeled according to the illustration shown in Figure 17. Inputs are indicated by the tail of an arrow and the outputs are labeled with an arrow head. The inputs are identified by a signal name followed by a source figure number. The outputs are identified by a signal name followed by a destination figure number.

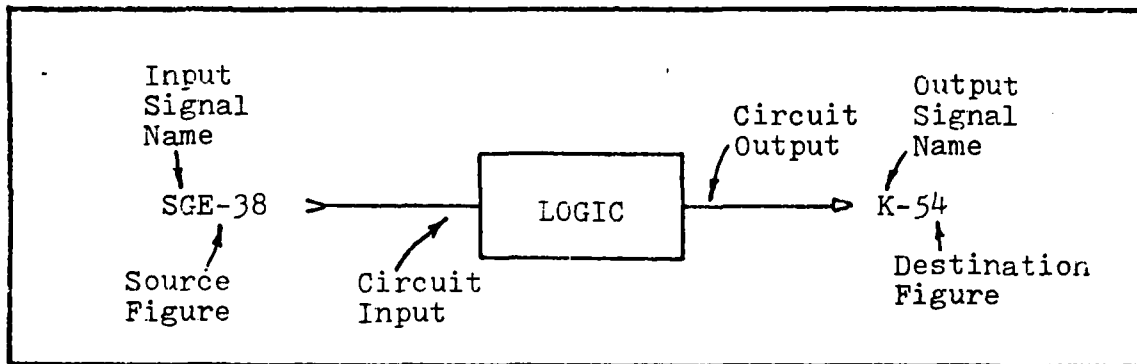


Fig. 17. Signal Labeling

This signal naming system is referred to as the Combination System in the AFIT Digital Engineering Laboratory Manual, Appendix 3. This system is used so that all signals can be readily traced throughout the entire system. The logic elements that do not have a number assigned have not been incorporated into the system.

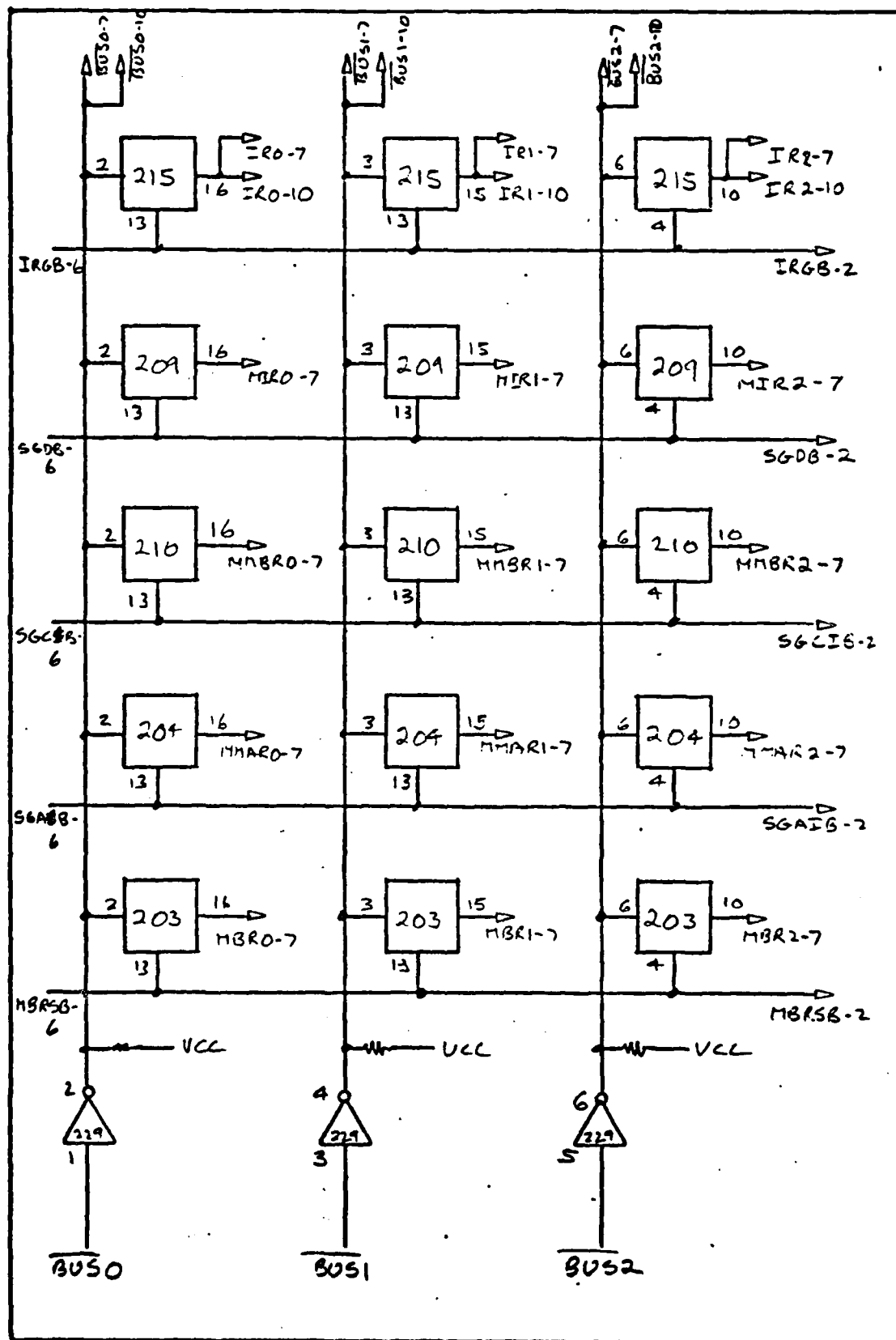


Fig. 1. Signal Synthesizer (Bits 0-2)

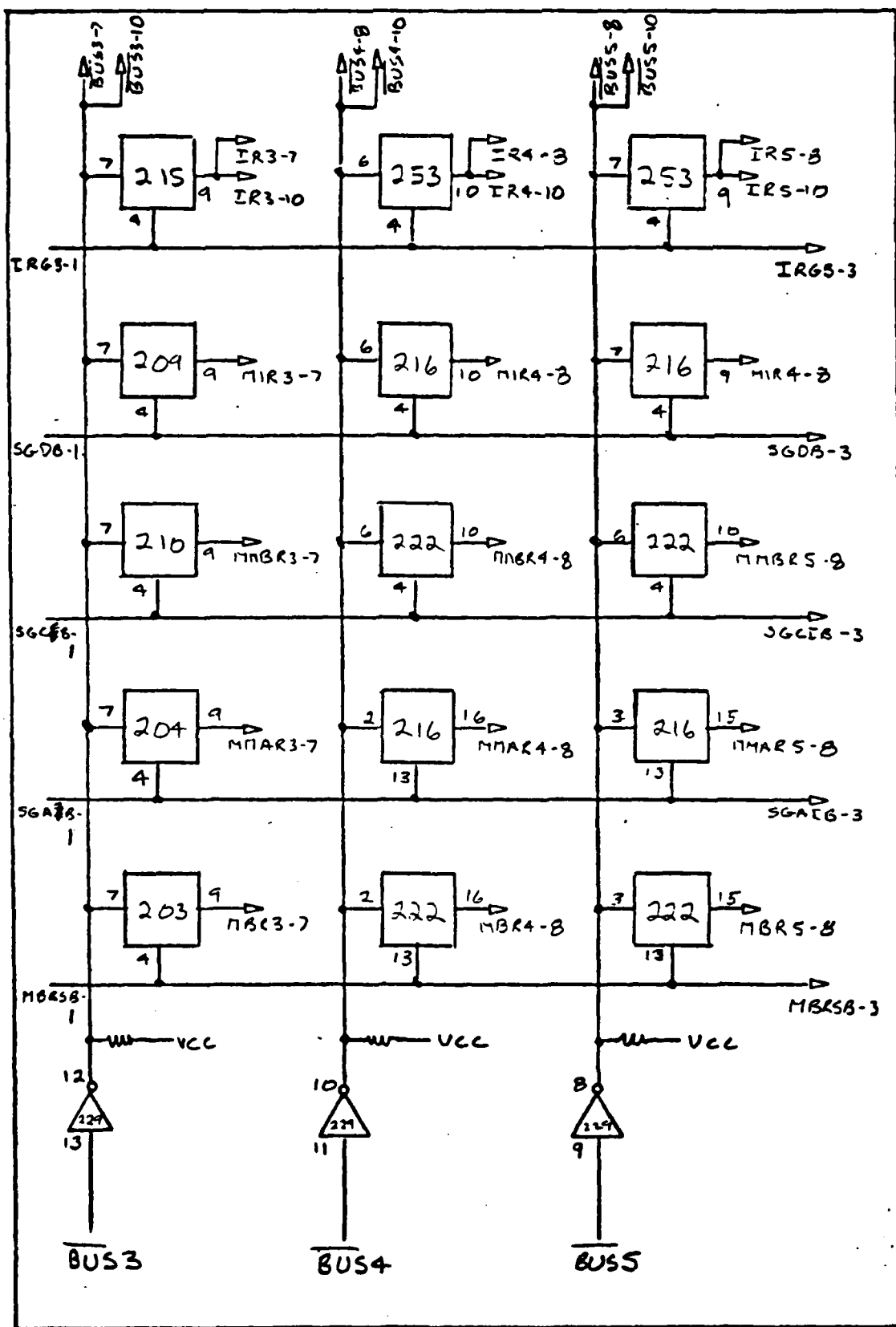


Fig. 2. Signal Synthesizer (Bits 3-5)

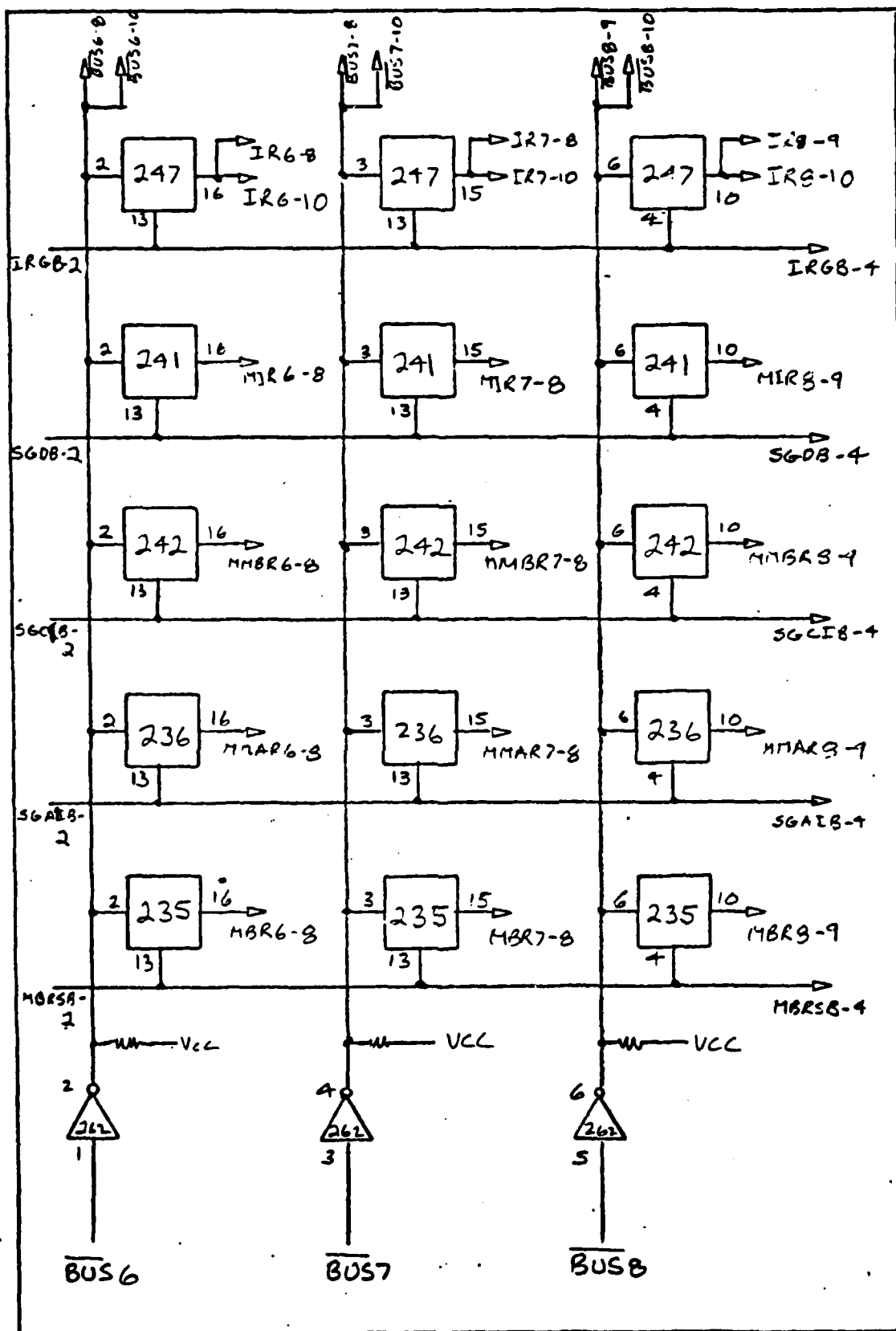


Fig. 3. Signal Synthesizer (Bits 6-8)

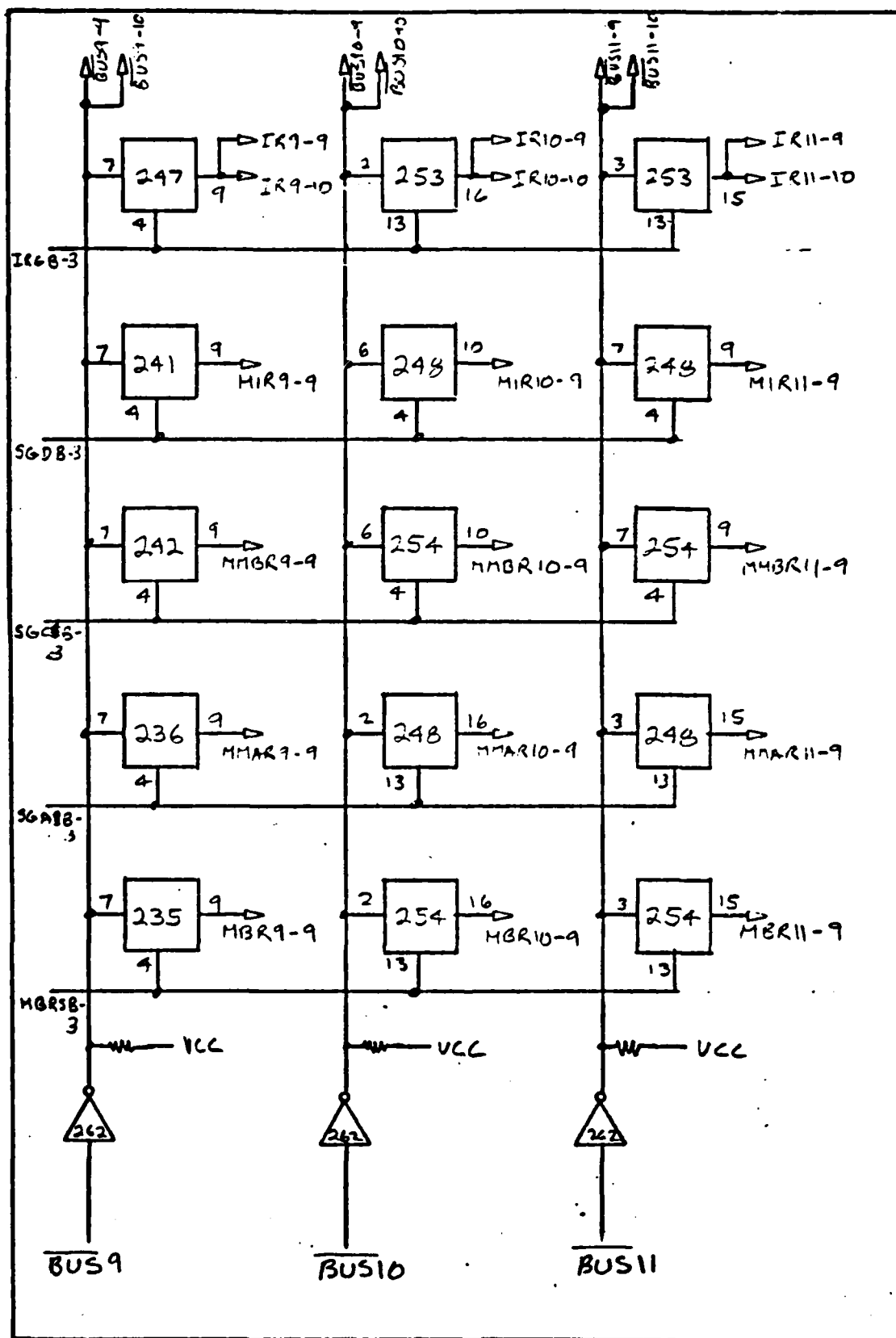


Fig. 4. Signal Synthesizer (Bits 9-11)

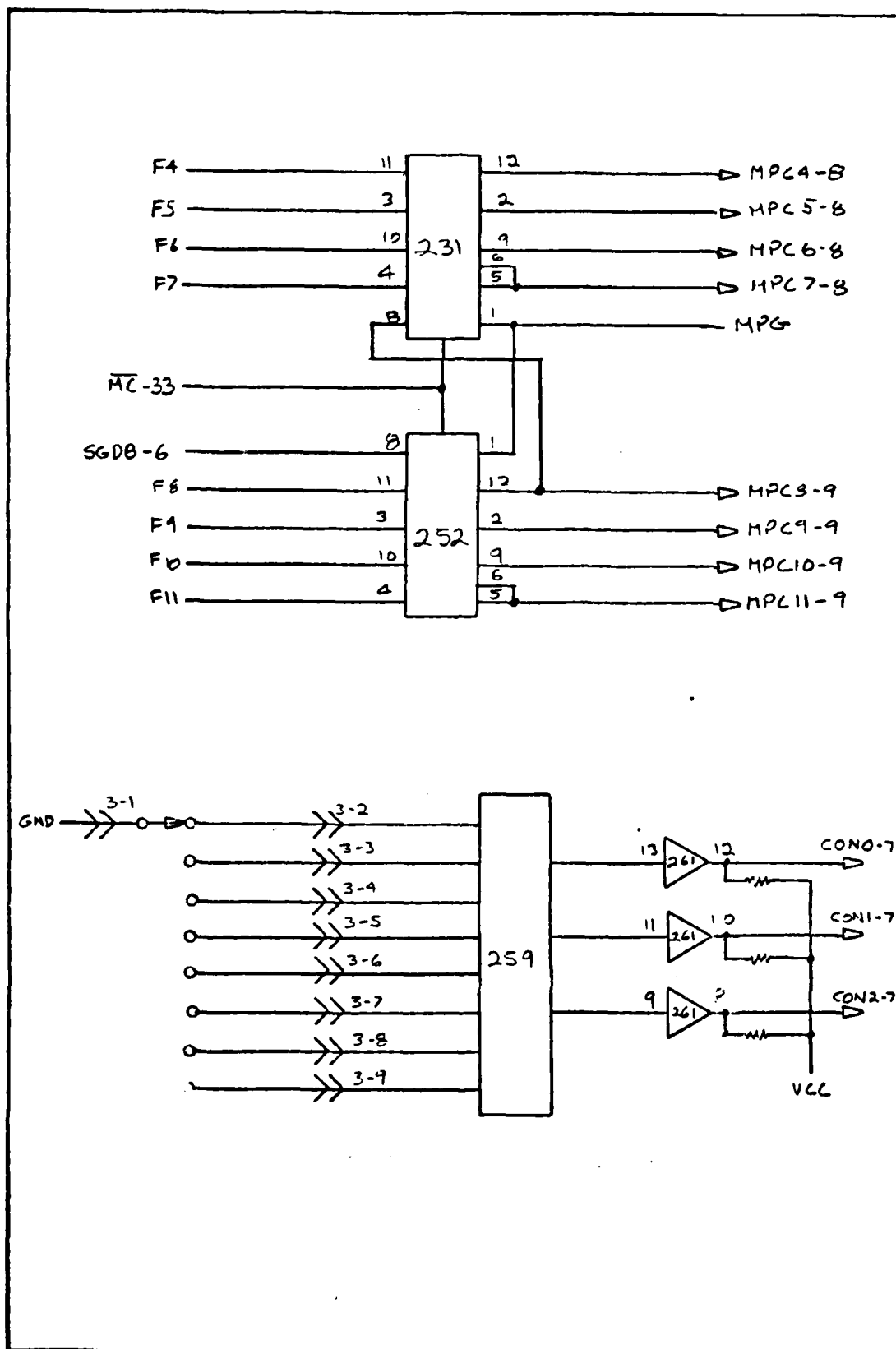
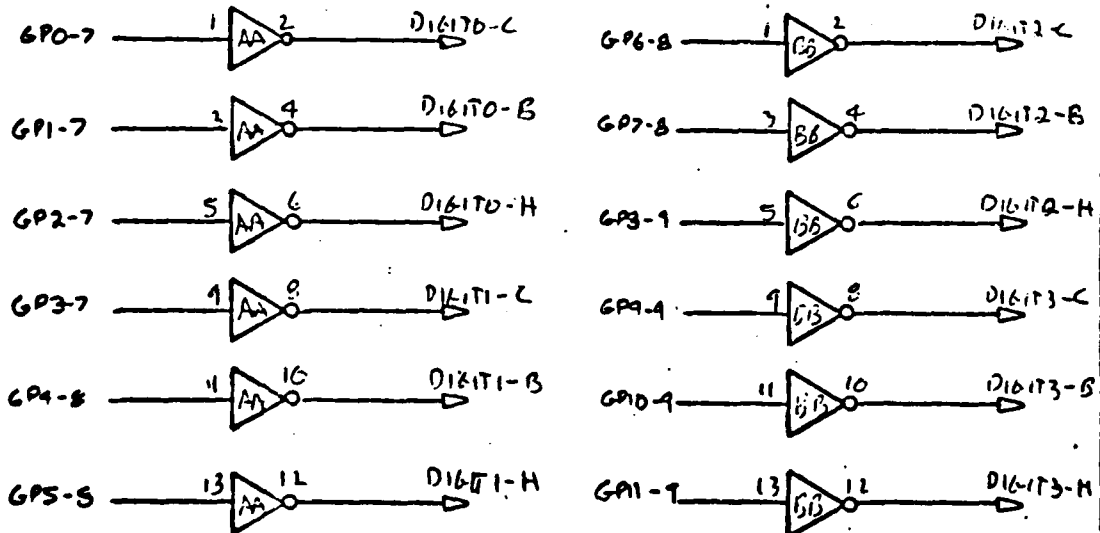
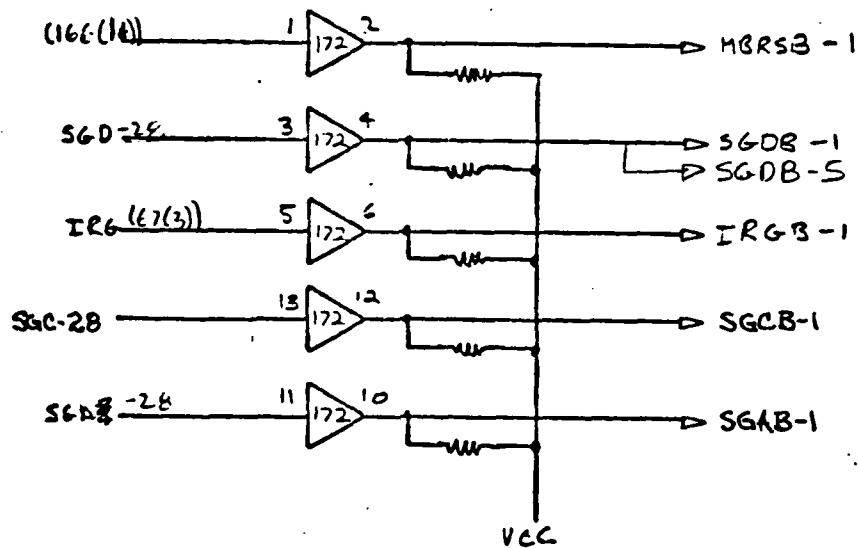


Fig 5. MPC Synthesizer and Encoder



NOTE: AA and BB = SN7404 located on LED Termination Board. Digit ( ) = Octal Display Digit

Fig. 6. Buffer Driver and GP Inverter

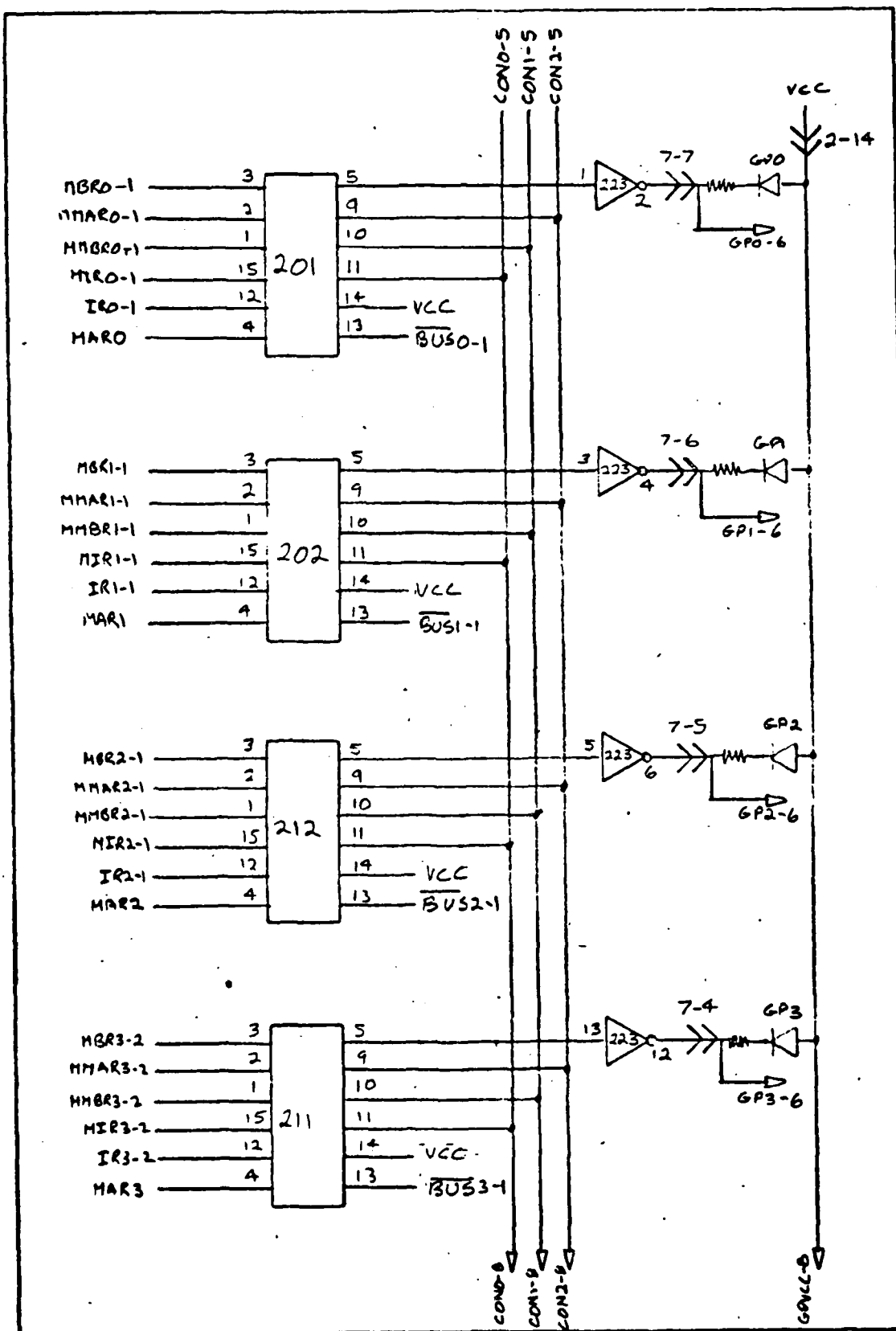


Fig. 7. Multiplexer (Bits 0-3)

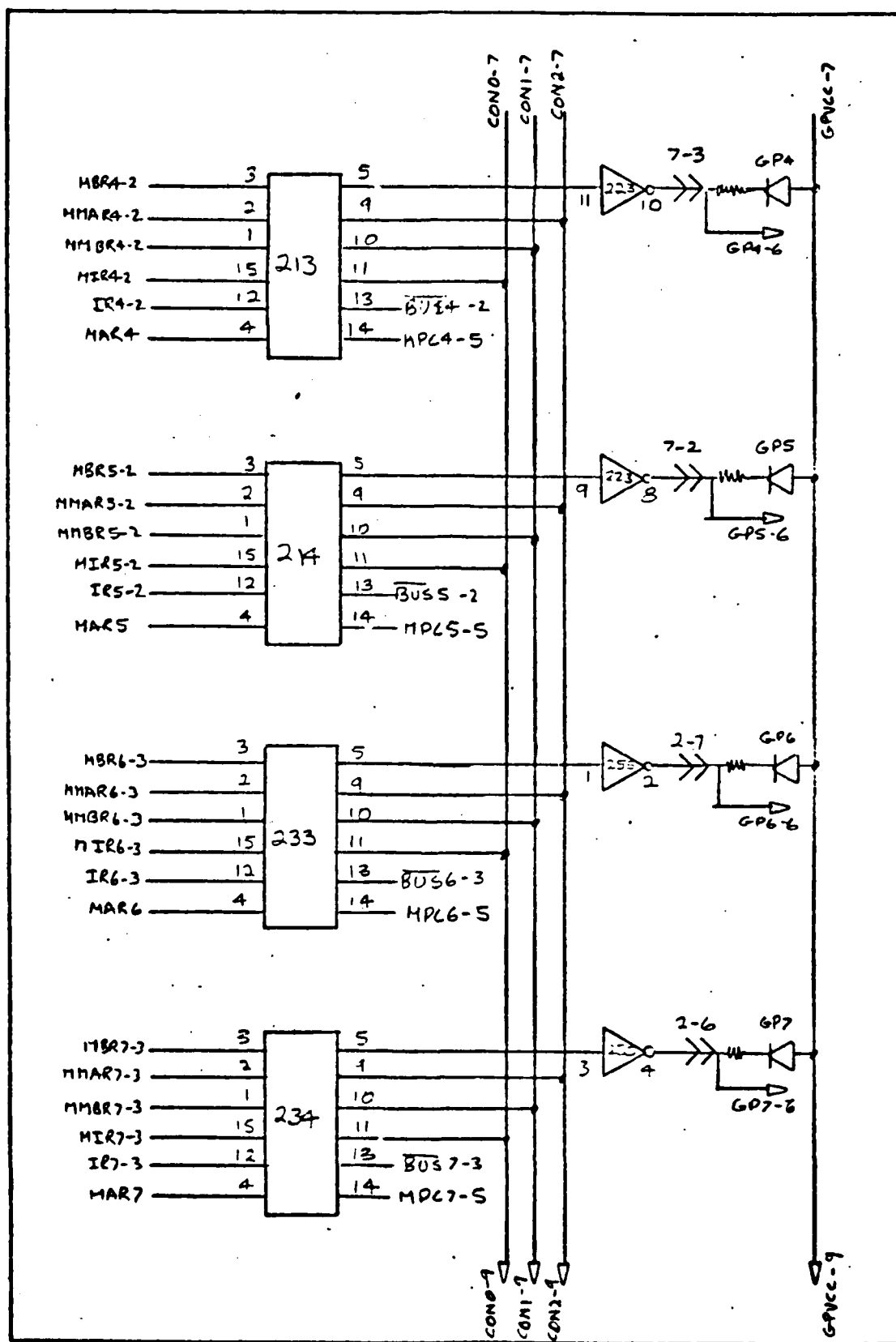


Fig. 8. Multiplexer (Bits 4-7)

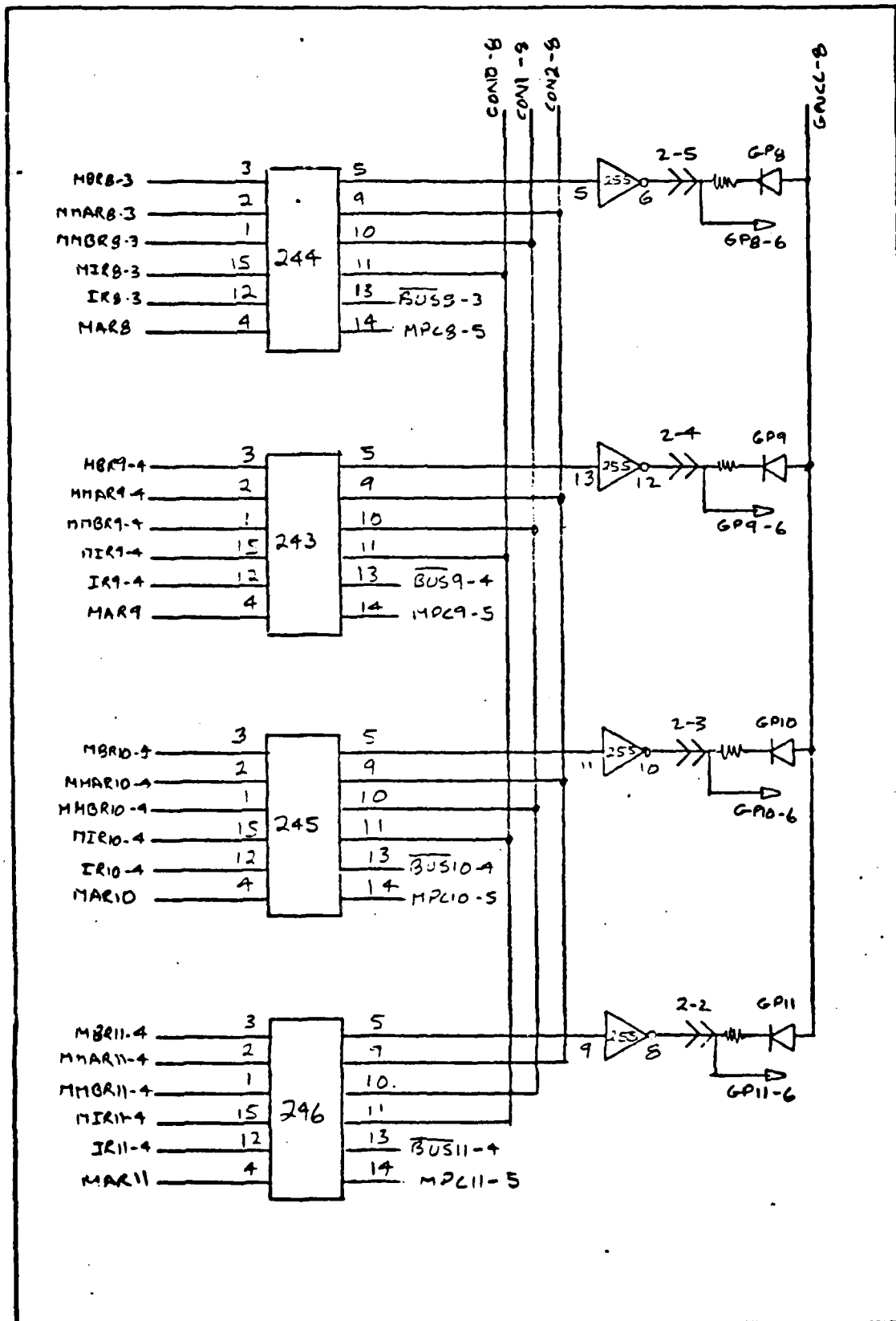


Fig. 9. Multiplexer (Bits 8-11)

AD-A080 394

AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC P/6 9/2  
PEDAGOG II REALIZATION.(U)

DEC 79 J S SHEENAN

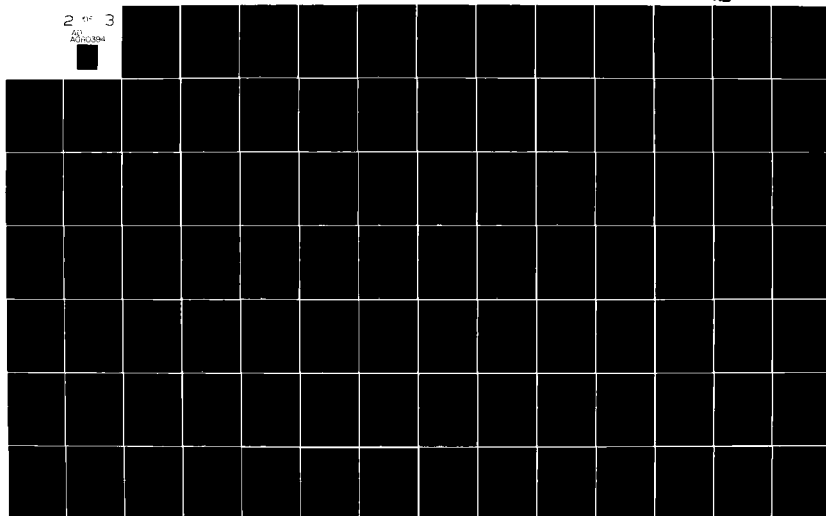
UNCLASSIFIED

AFIT/02/EE/79-033

NL

2 14 3

AD-A080394



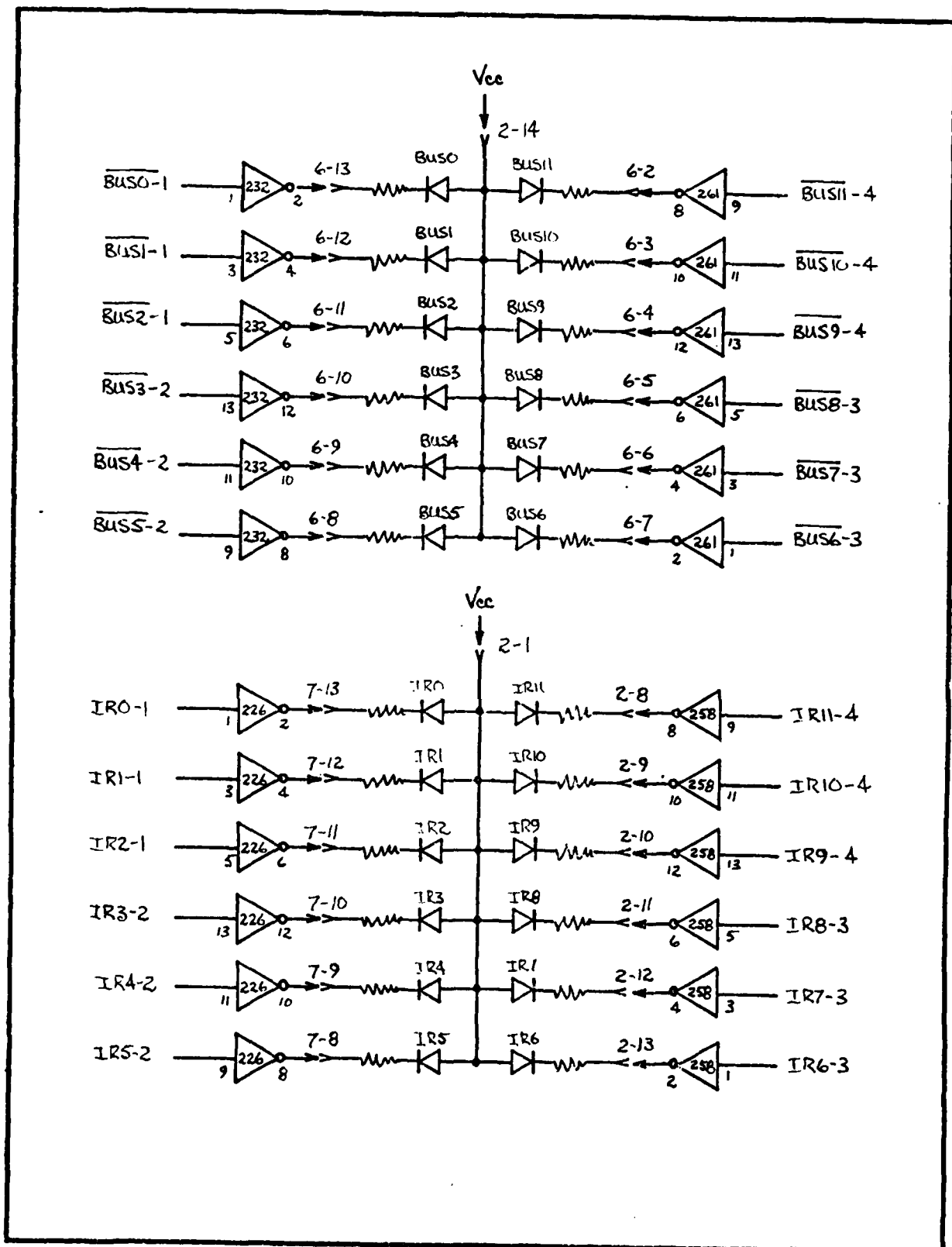


Fig. 10. BUS and IR Displays and Drivers

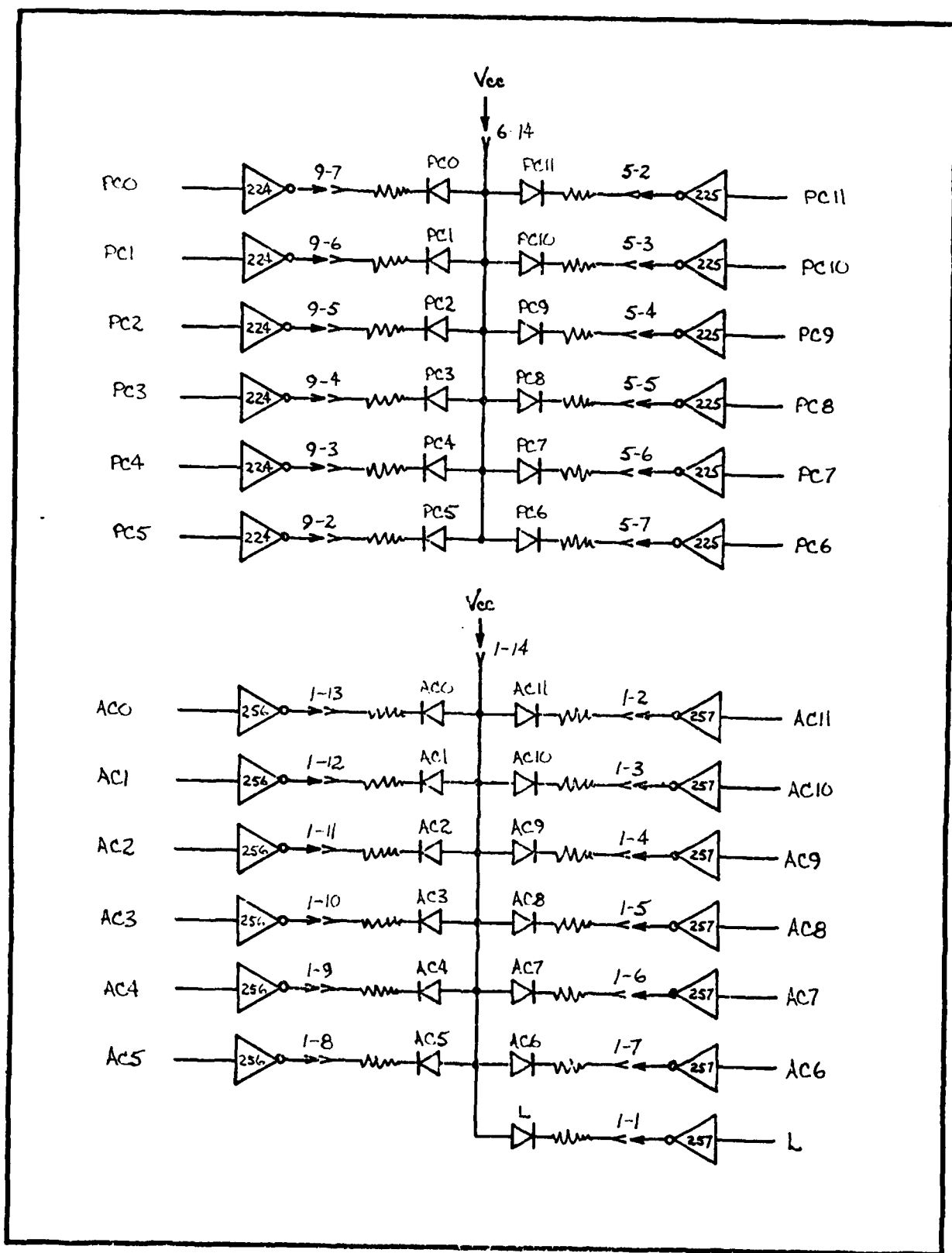


Fig. 11. PC and AC Displays and Drivers

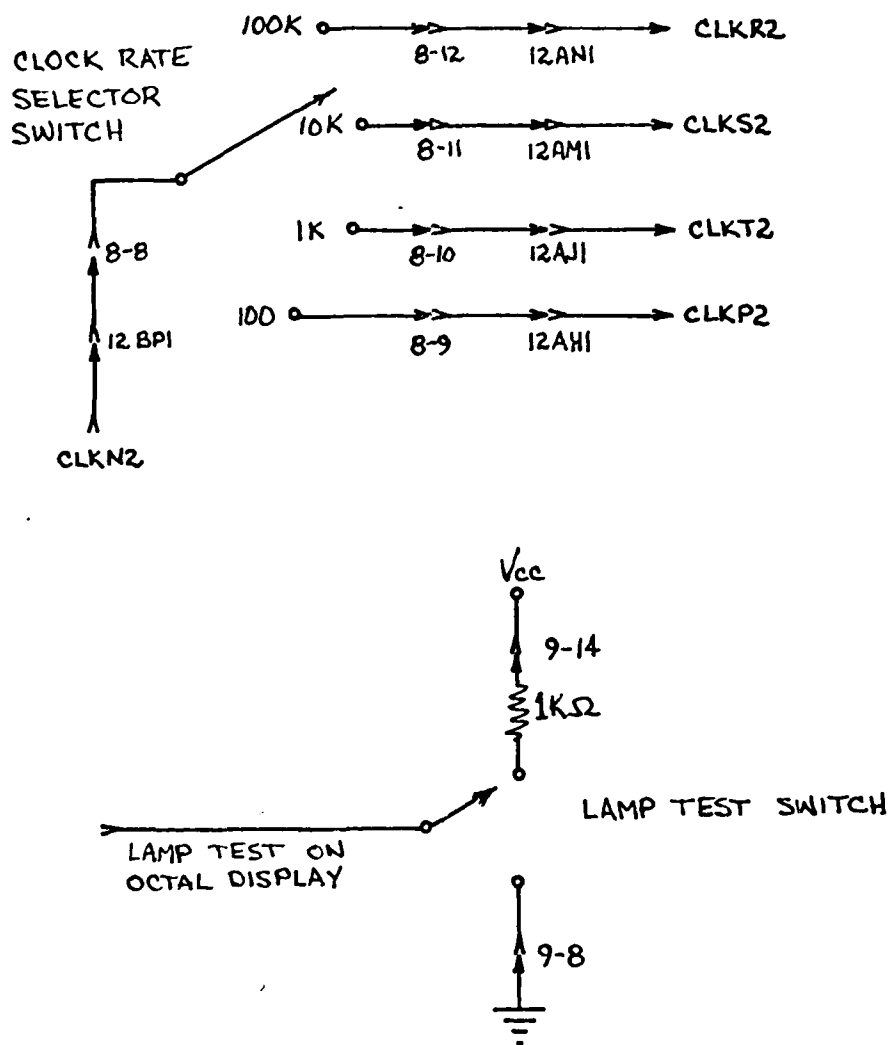


FIG. 12. Clock Rate Selector and Lamp Test

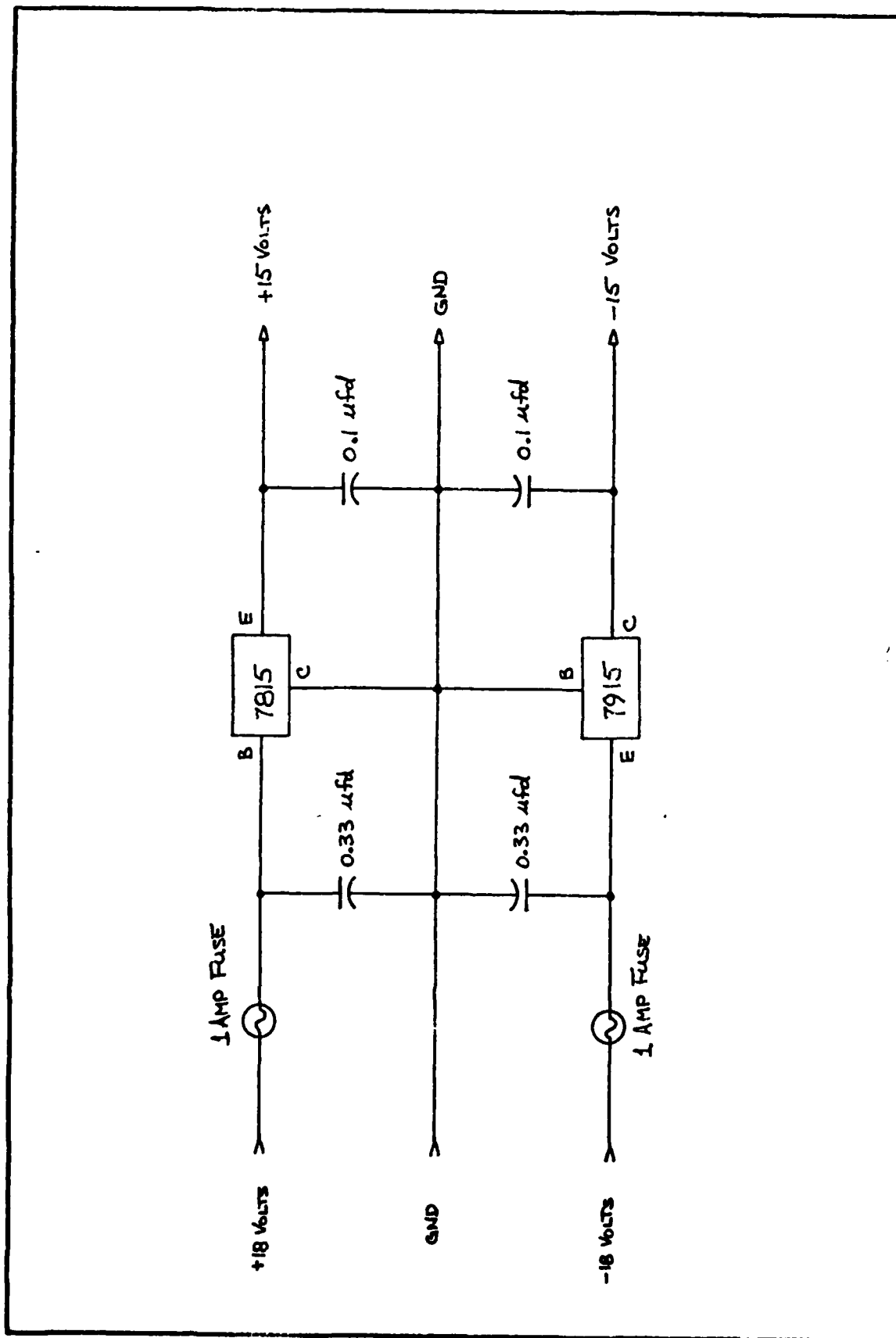


Fig. 20. Voltage Converter Circuit

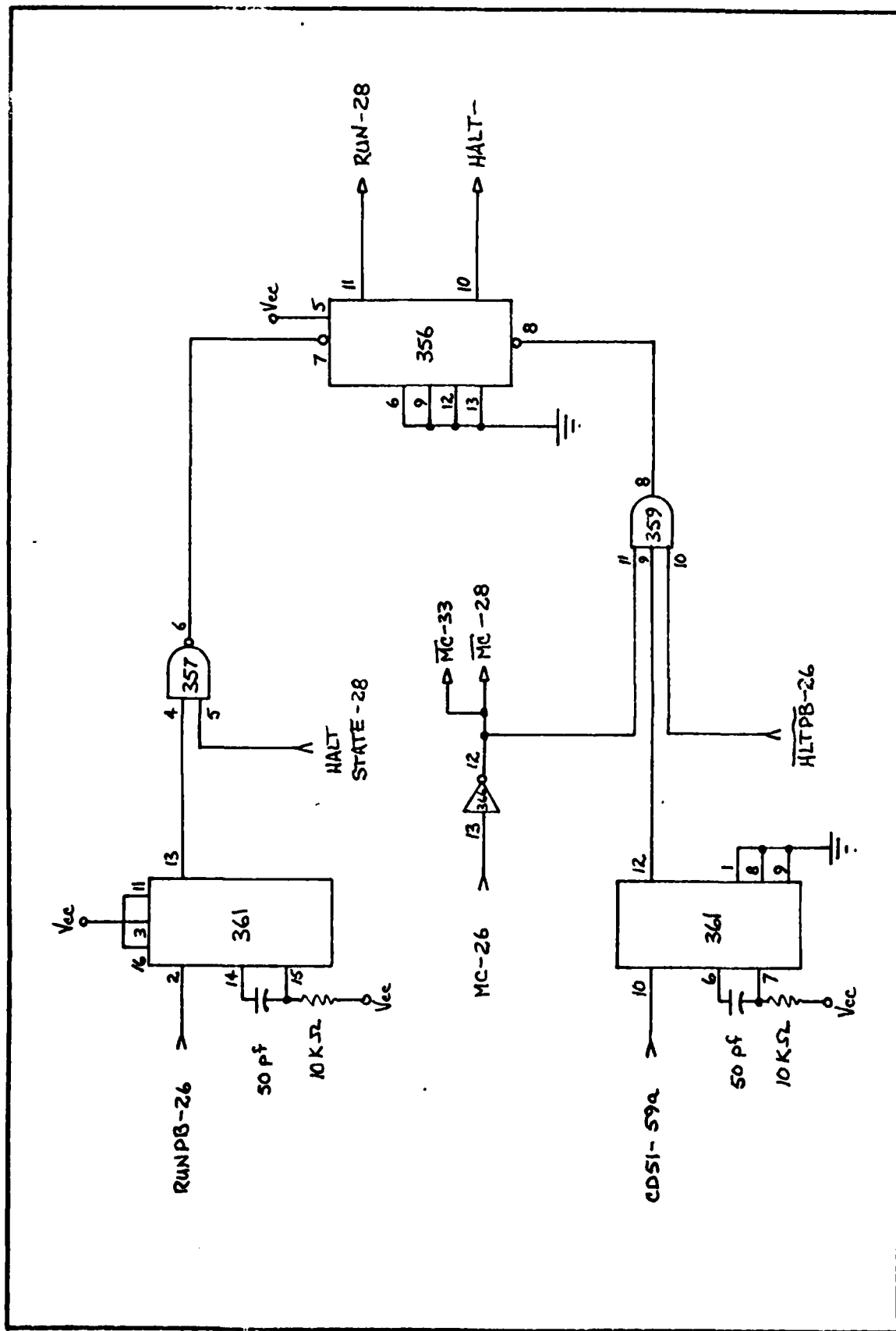


Fig. 23. Run/Halt Flip-Flop and Master Clear Circuit

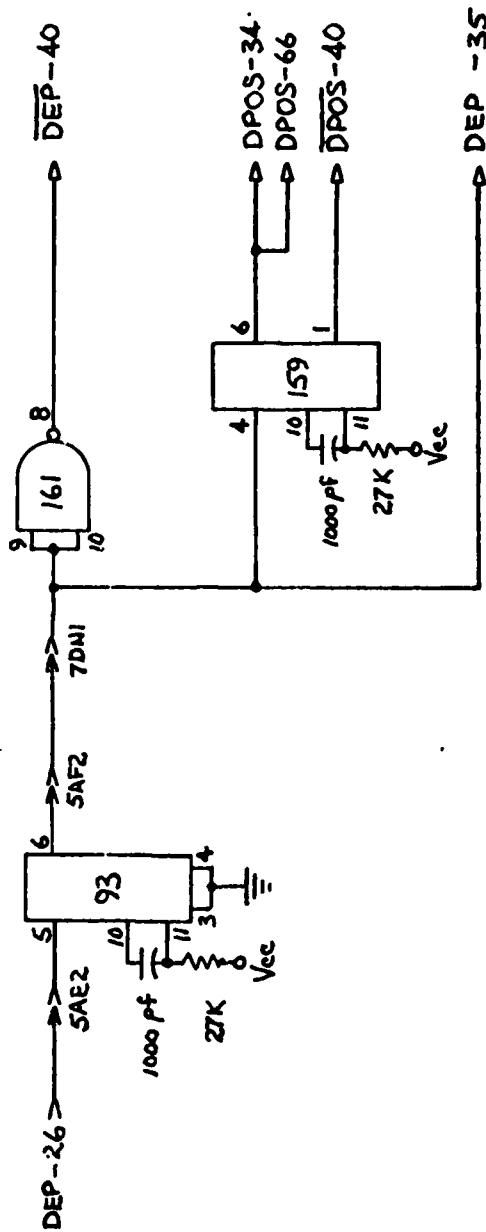


Fig. 24. Deposit Circuit



**Fig. 25. Examine and Memory Control Circuit**

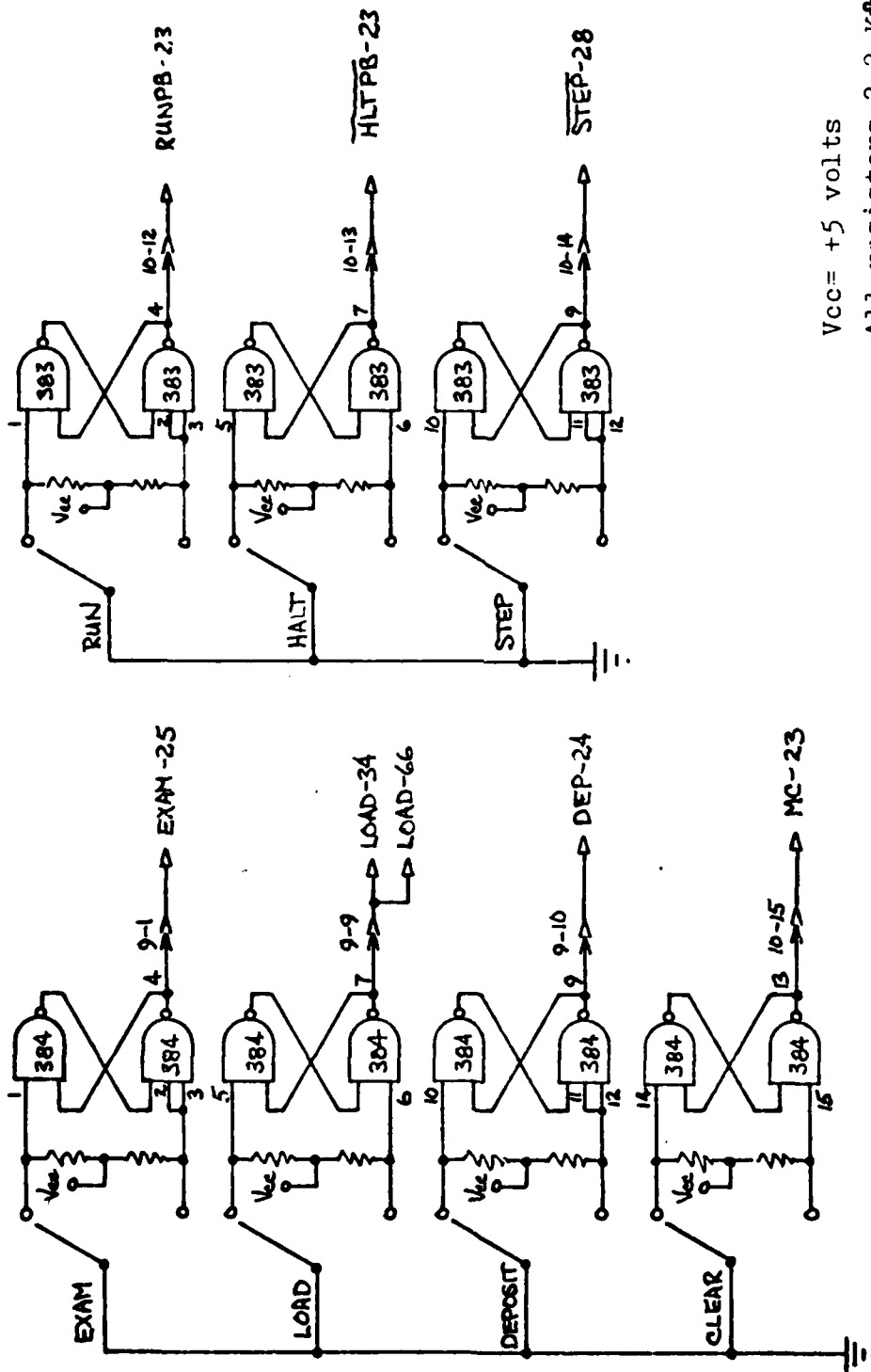


Fig. 26. Control Switch Debounce Circuits

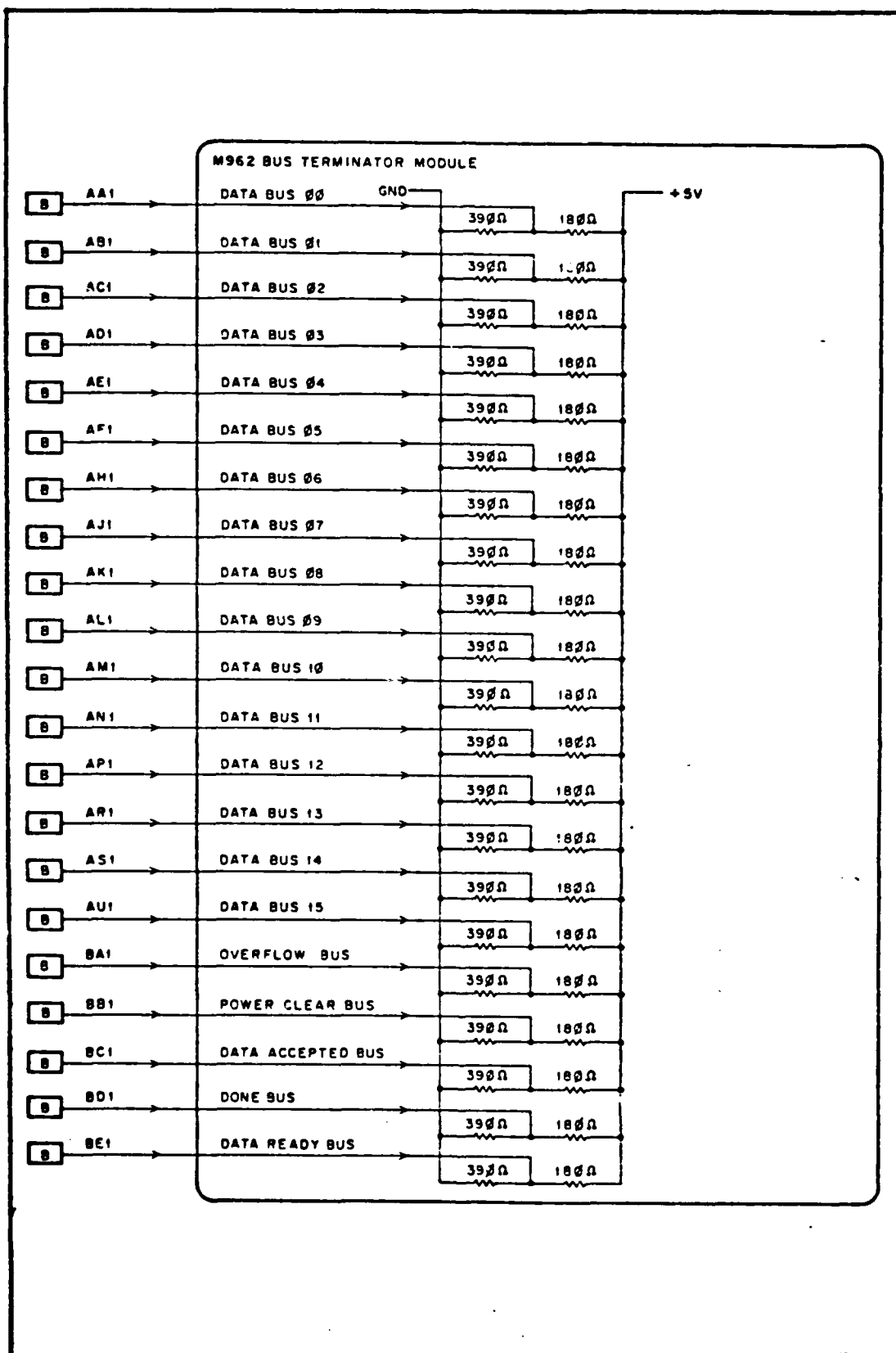


Fig. 27. Bus Terminator

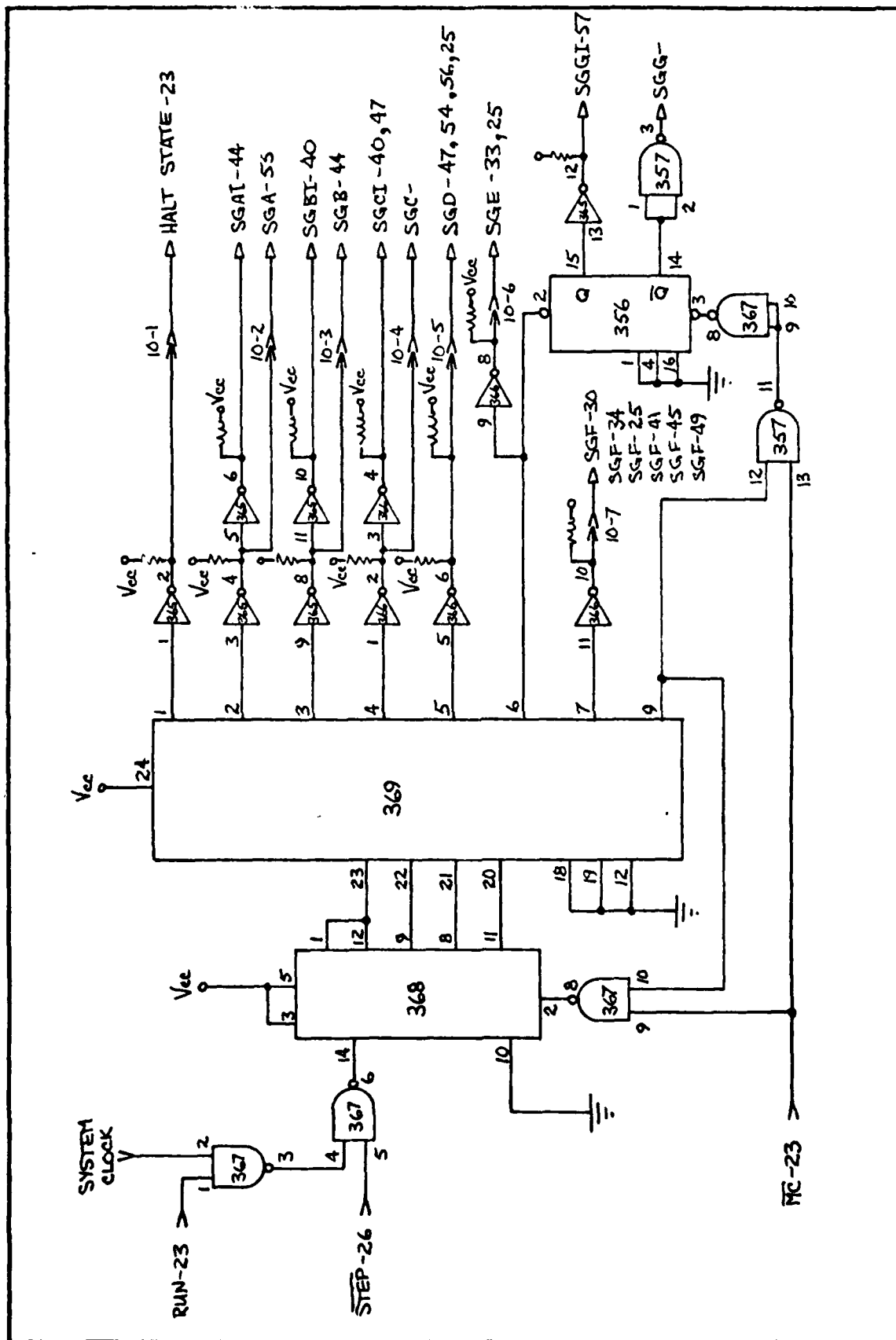


Fig. 28. Control State Generator

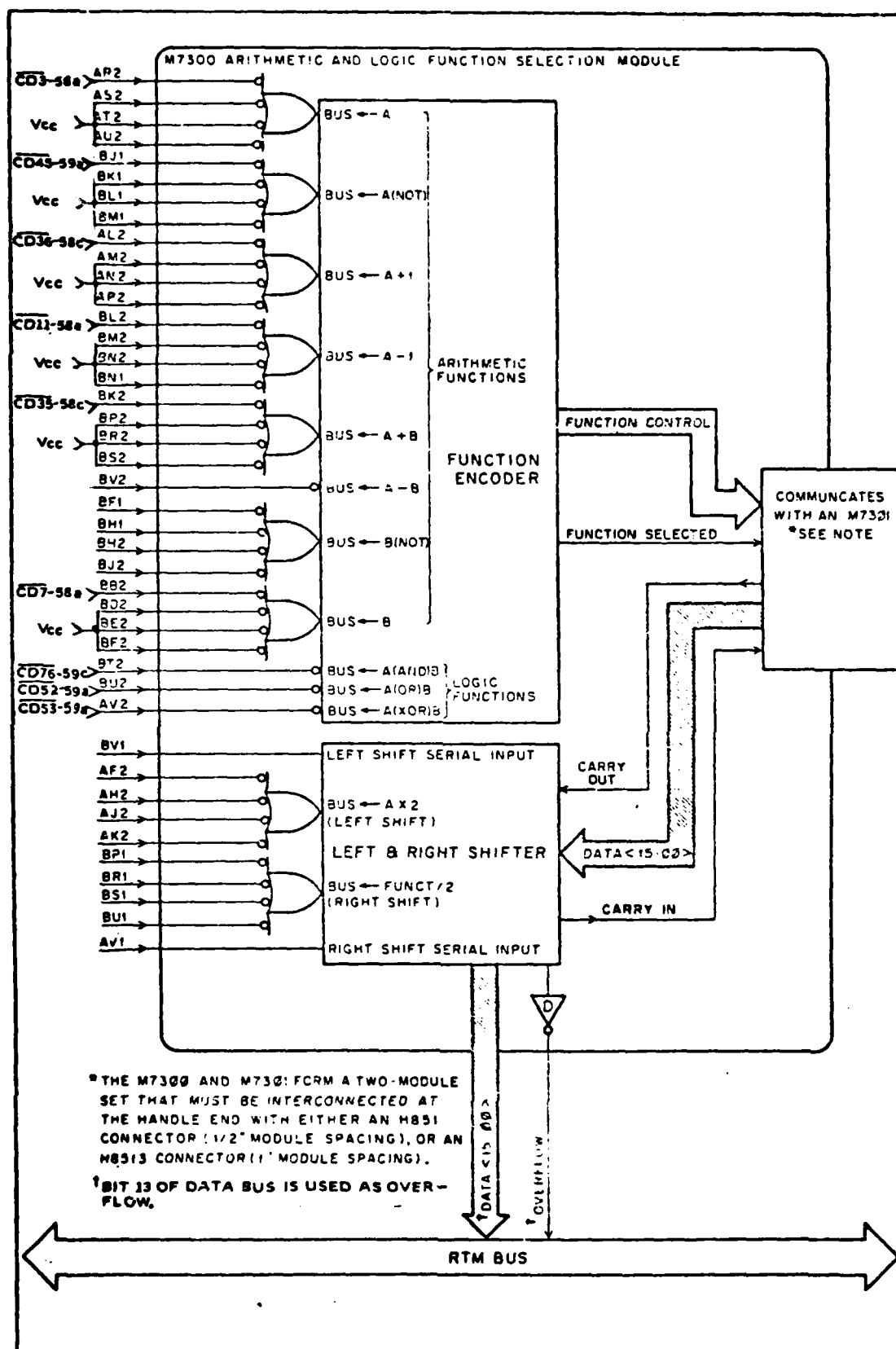


Fig. 29a. General Purpose Arithmetic Unit (M7300)

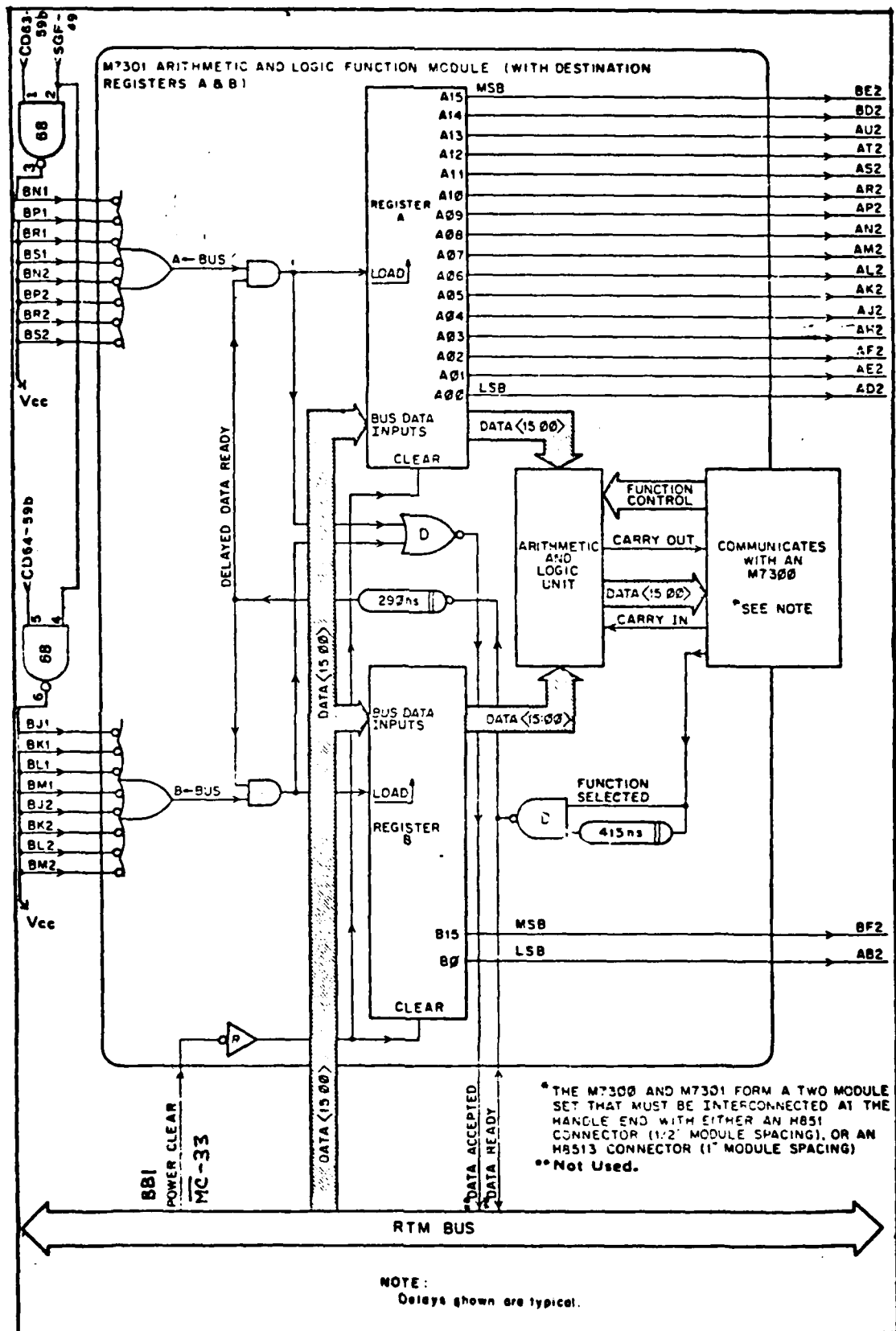


Fig. 29b General Purpose Arithmetic Unit (M7301)

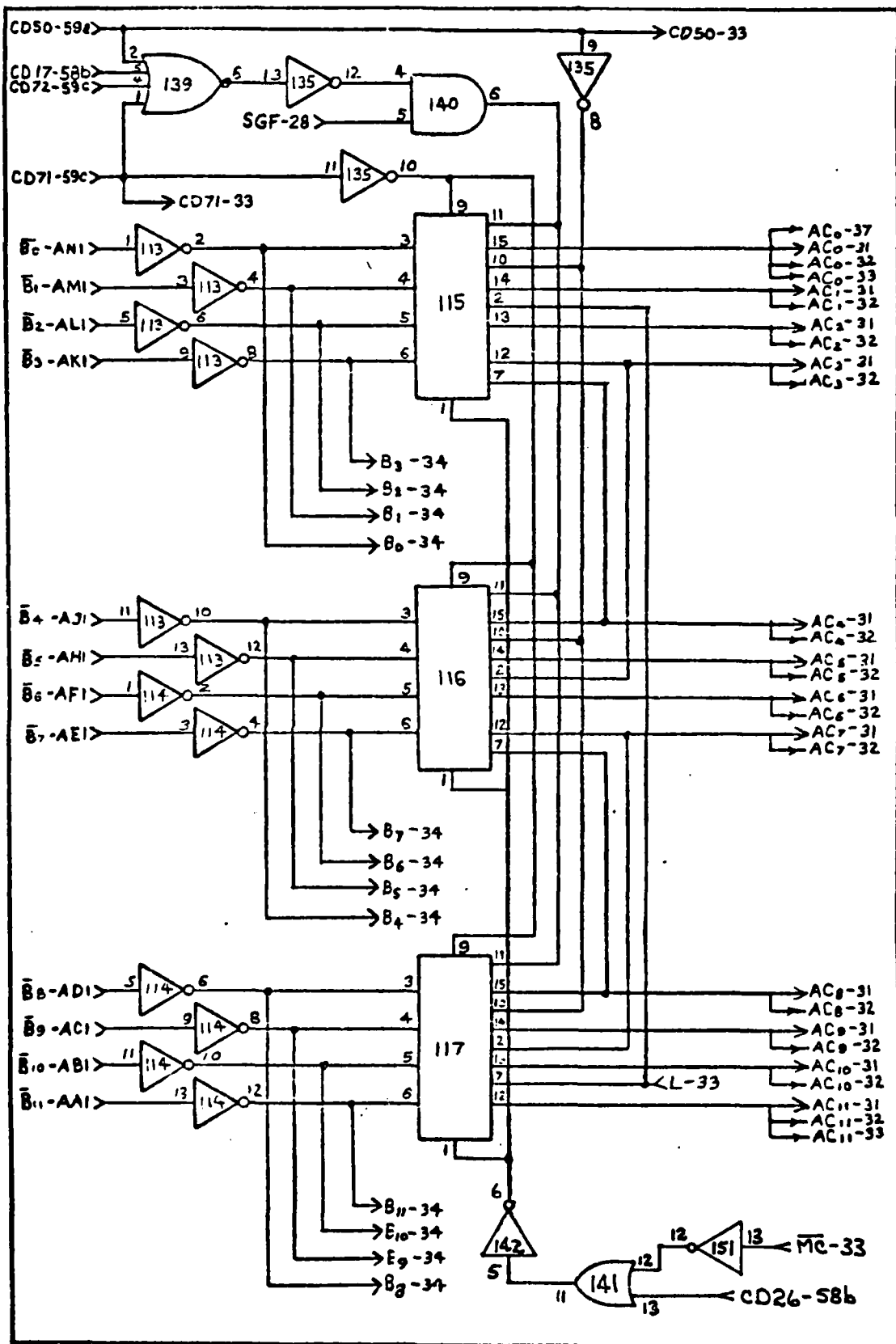


Fig. 30. Accumulator

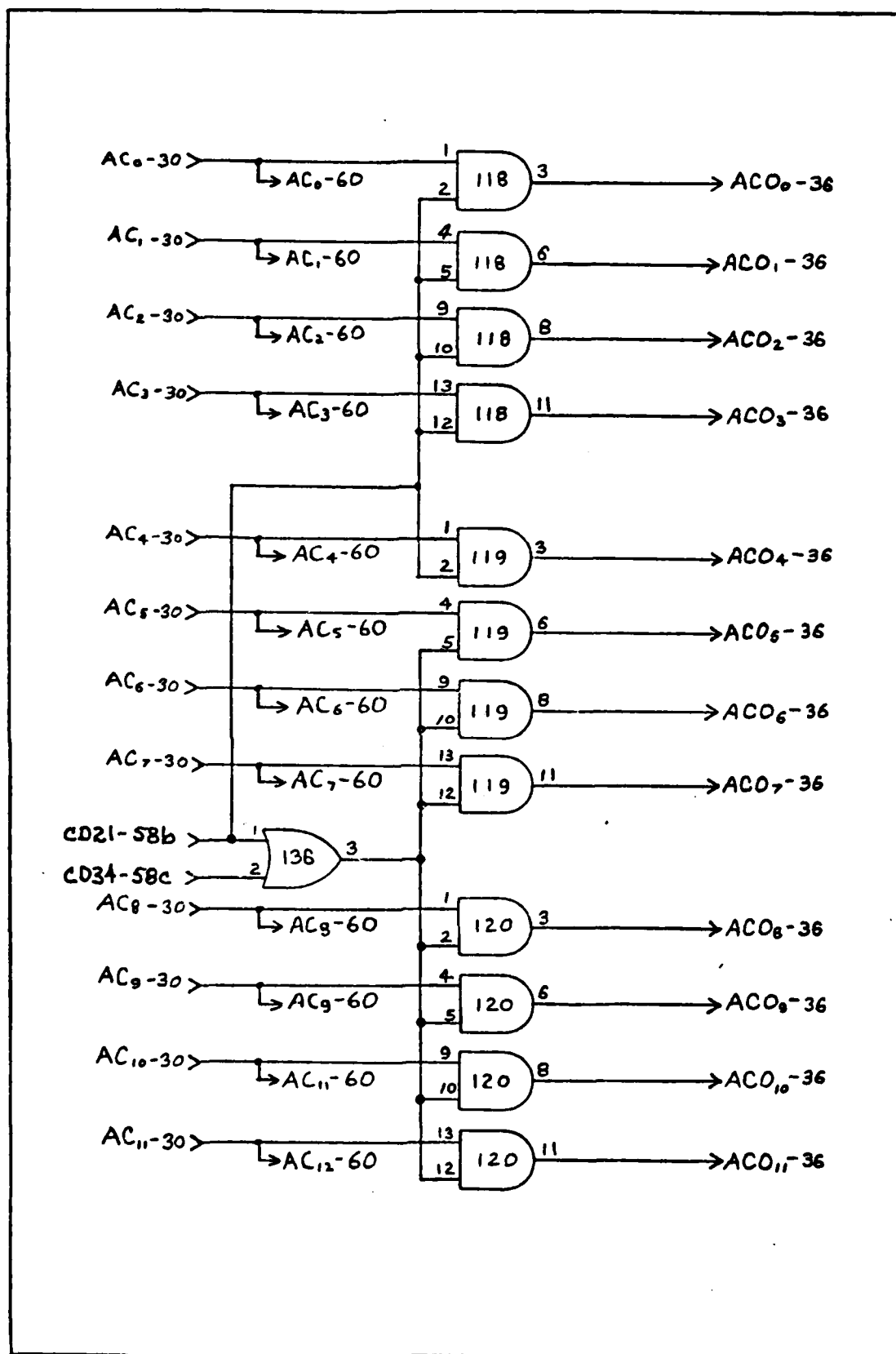


Fig. 31. Accumulator Output Switch

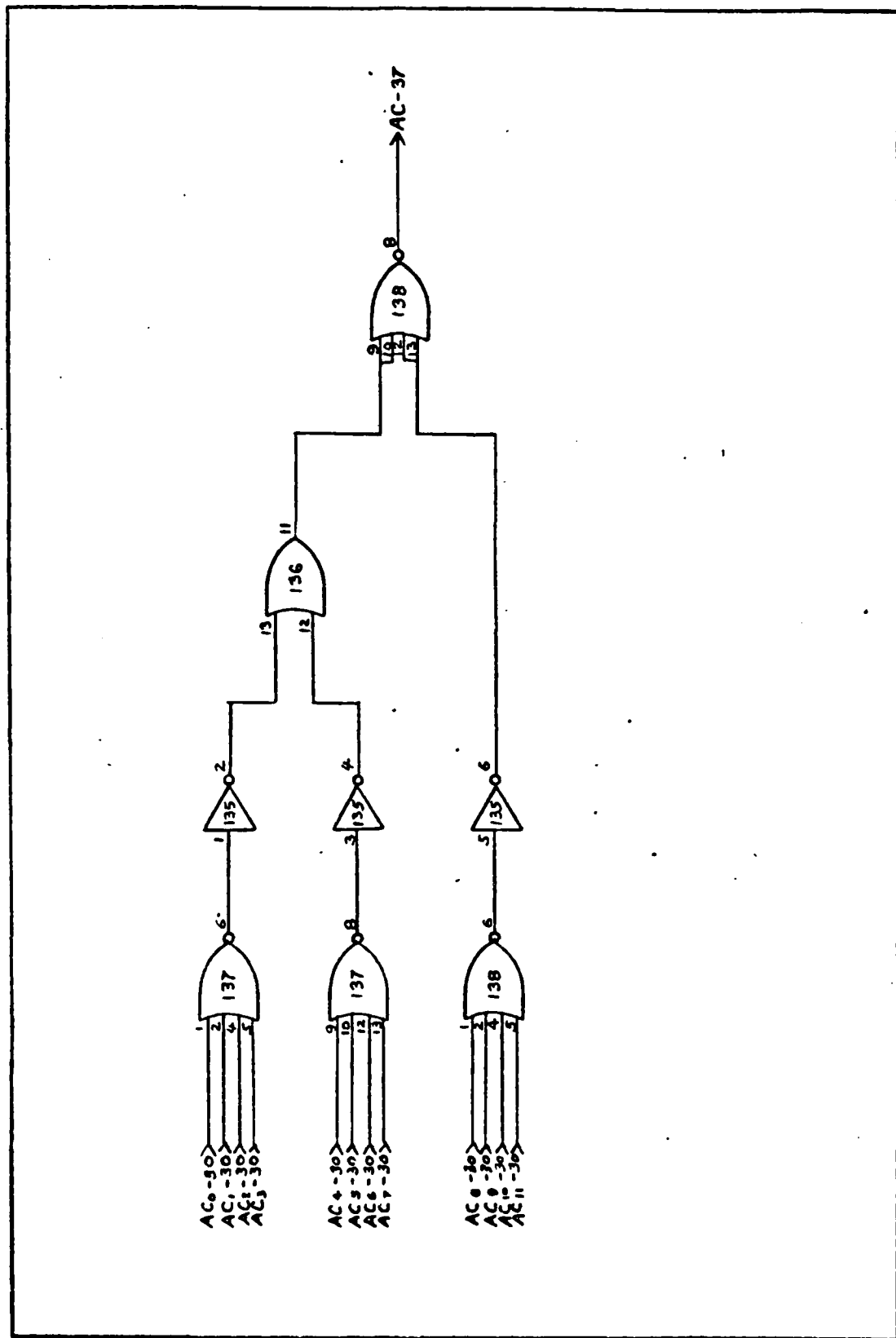


Fig. 32. Accumulator Status



97

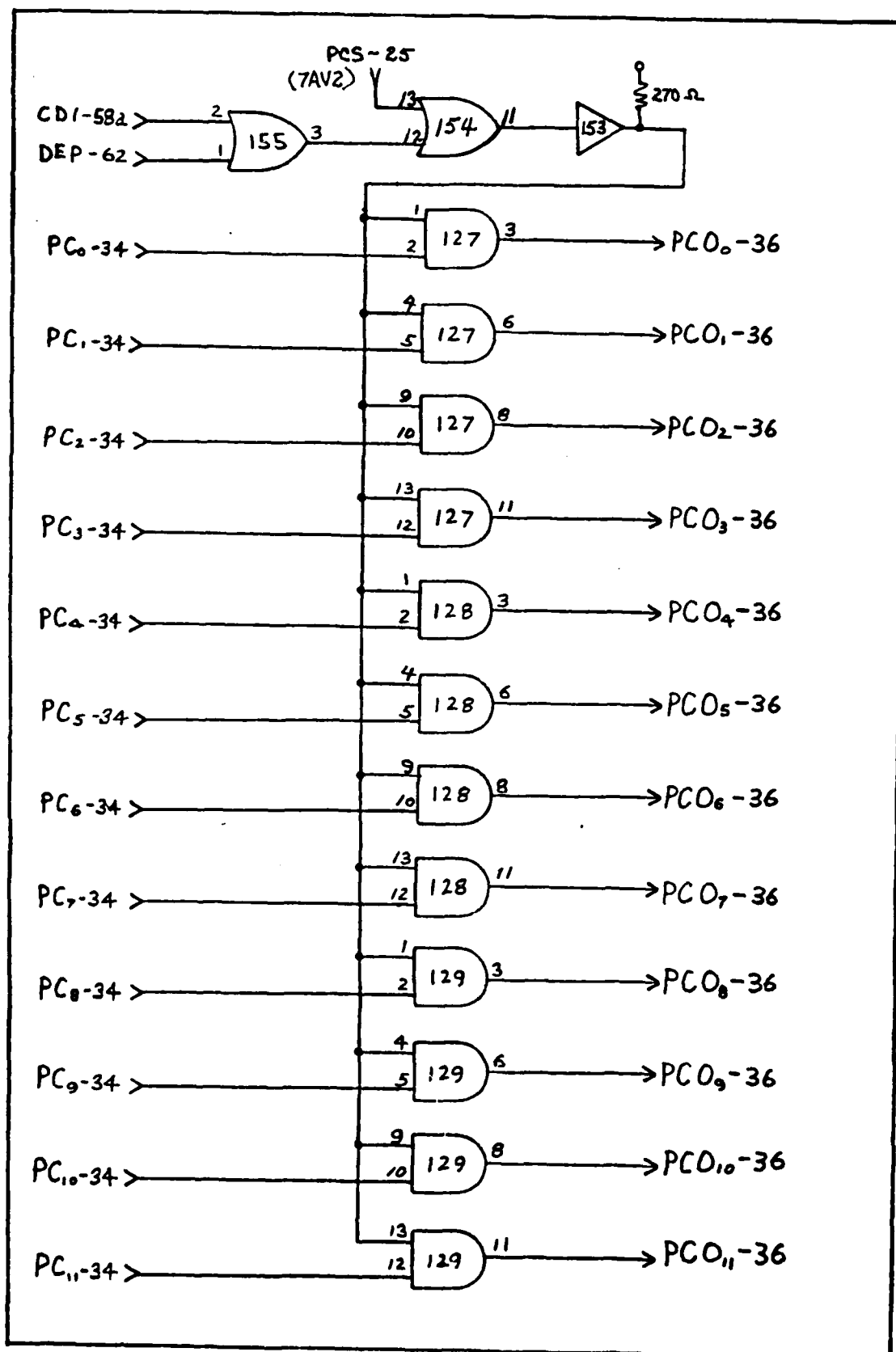


Fig. 35. Program Counter Output Switch

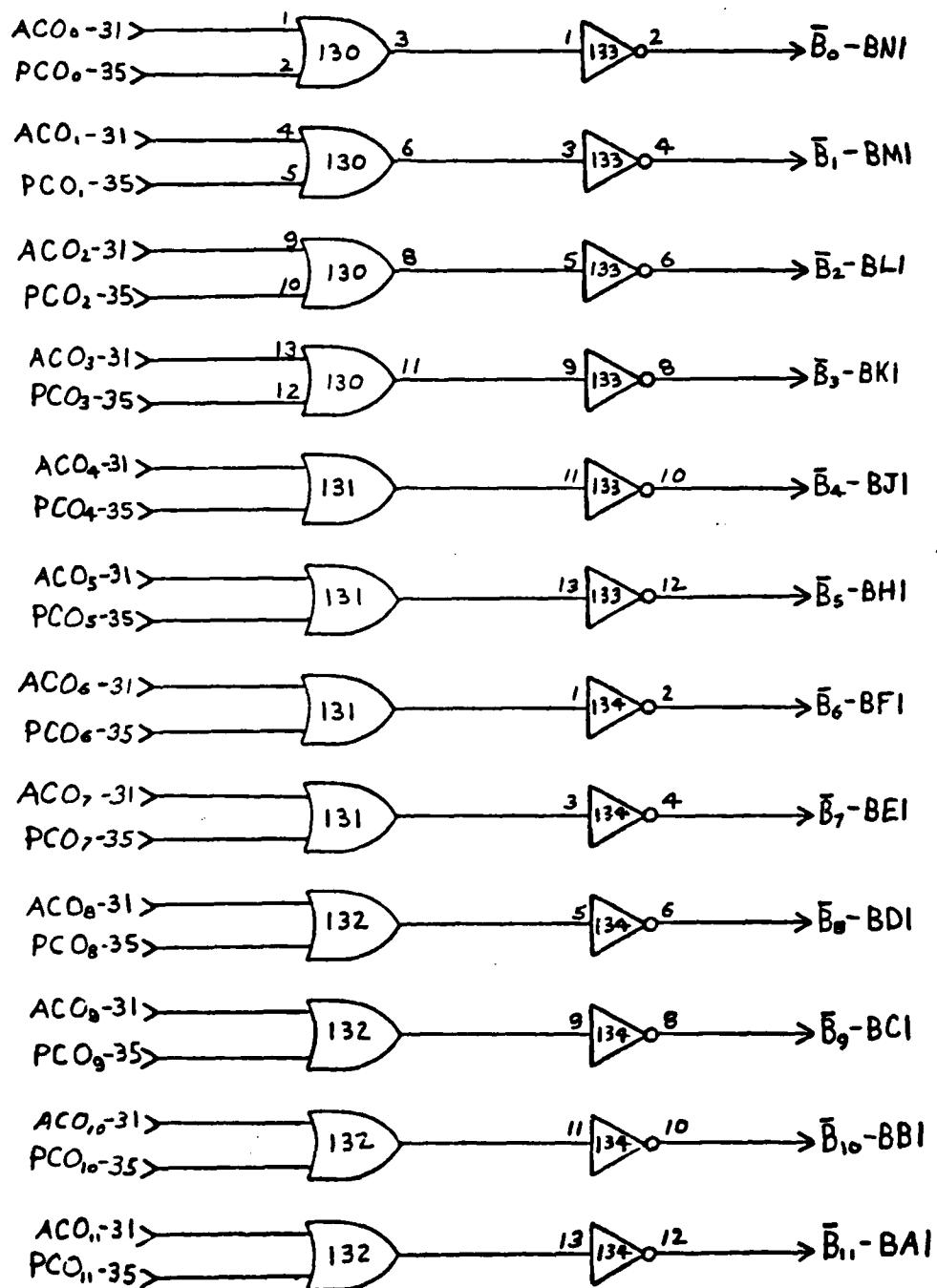


Fig. 36. Accumulator and Program Counter Bus Control Switch

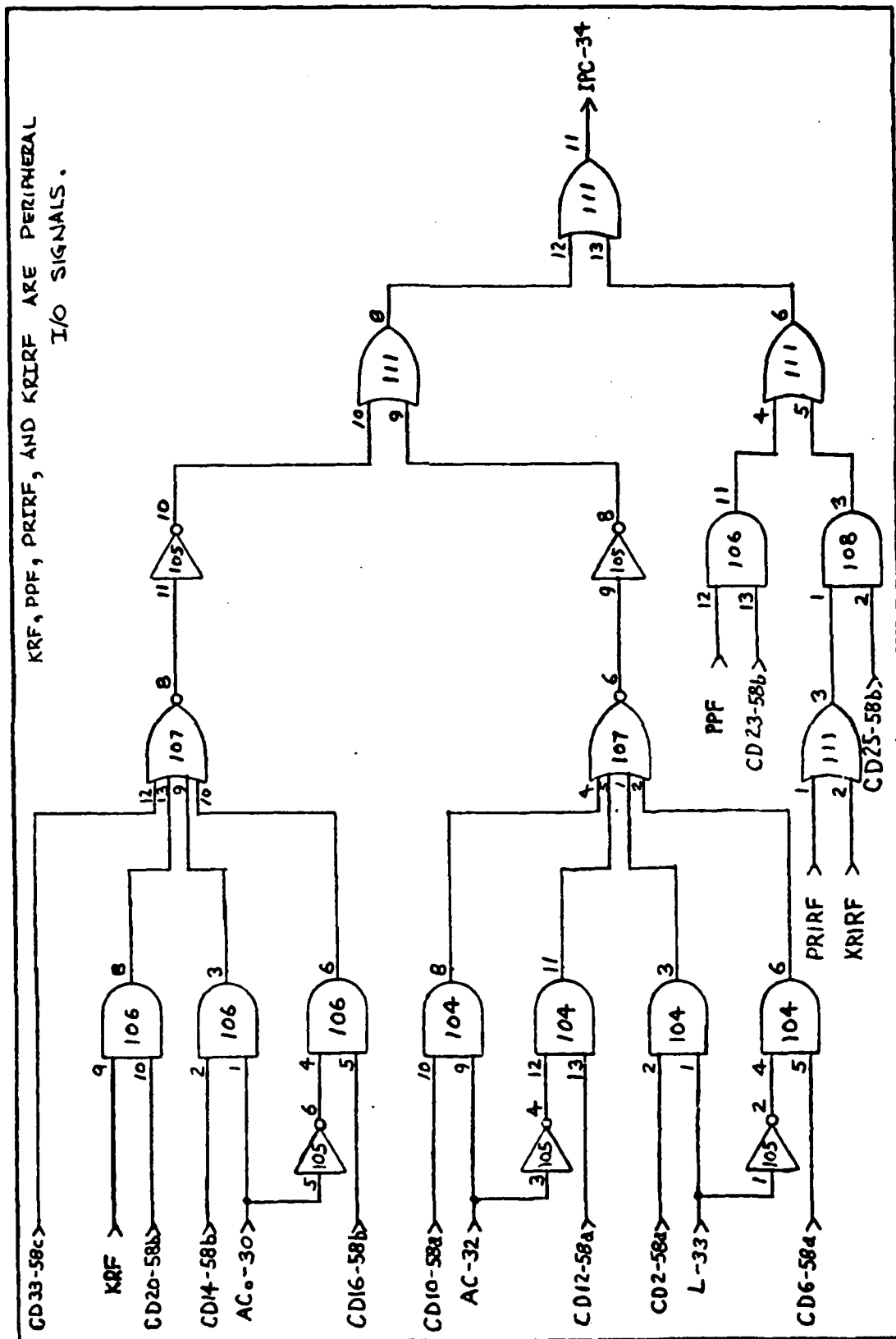
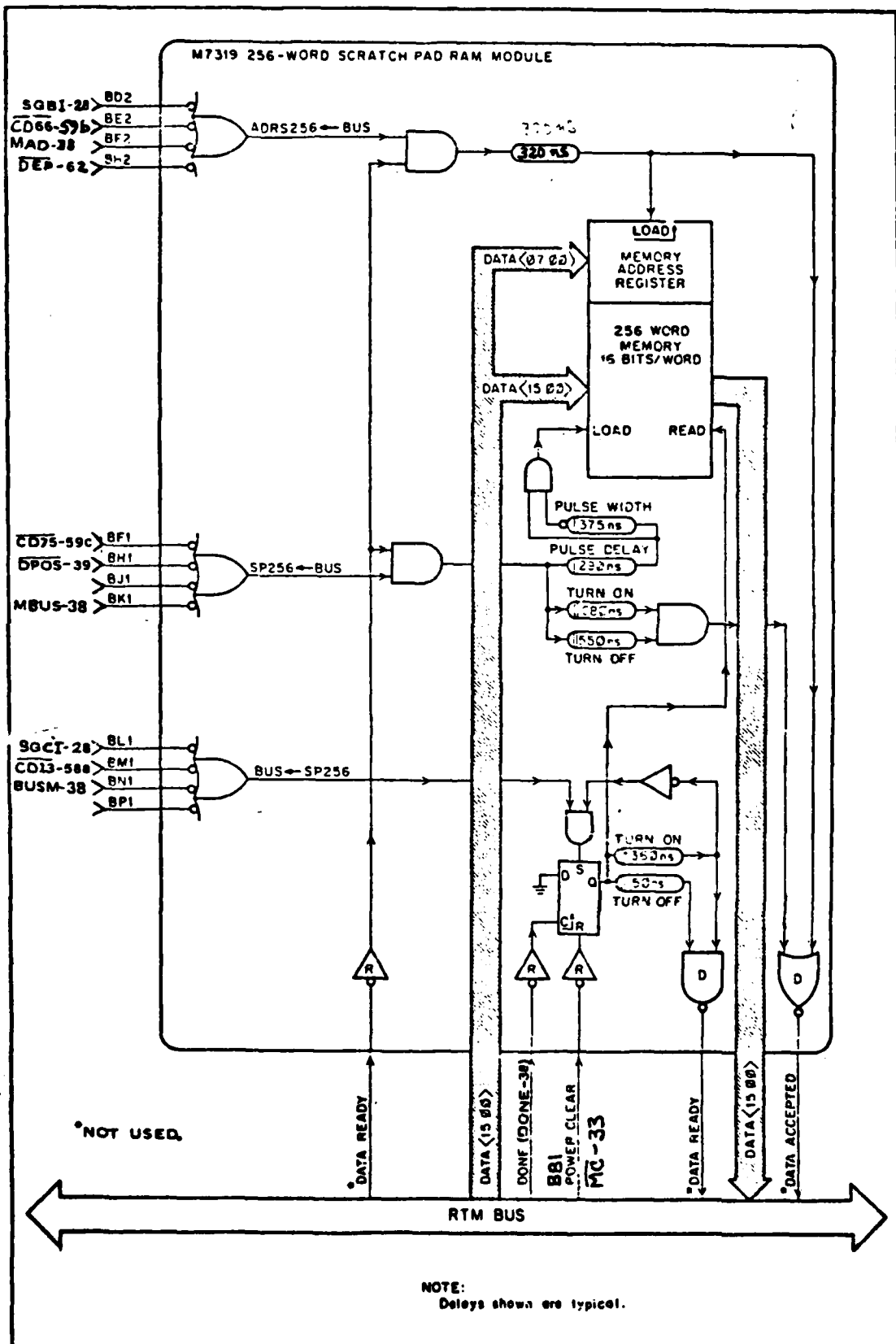


Fig. 37. Increment Program Counter



**Fig. 40. Memory Block Diagram**

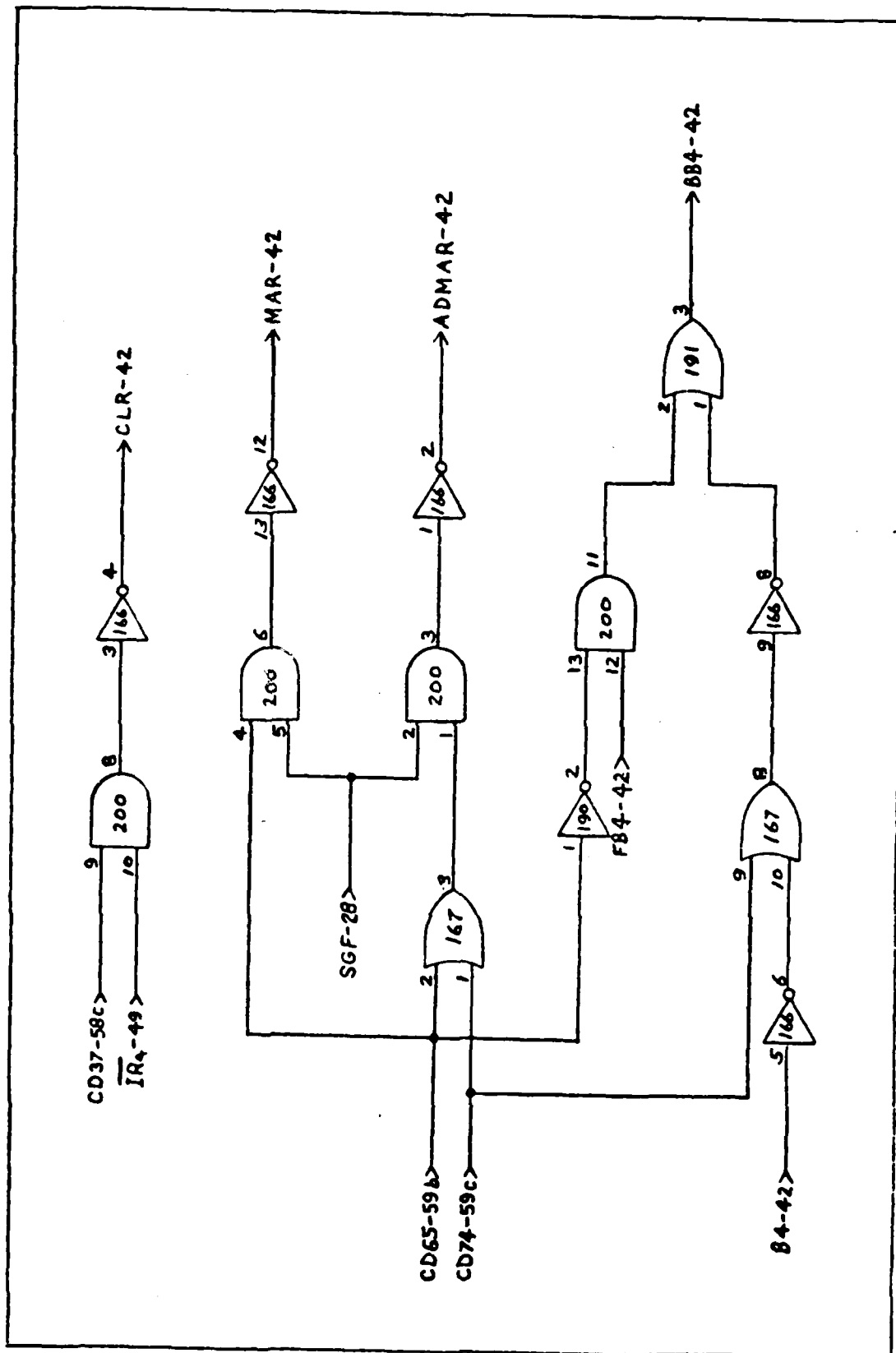


Fig. 41. Memory Address Register Control Circuit

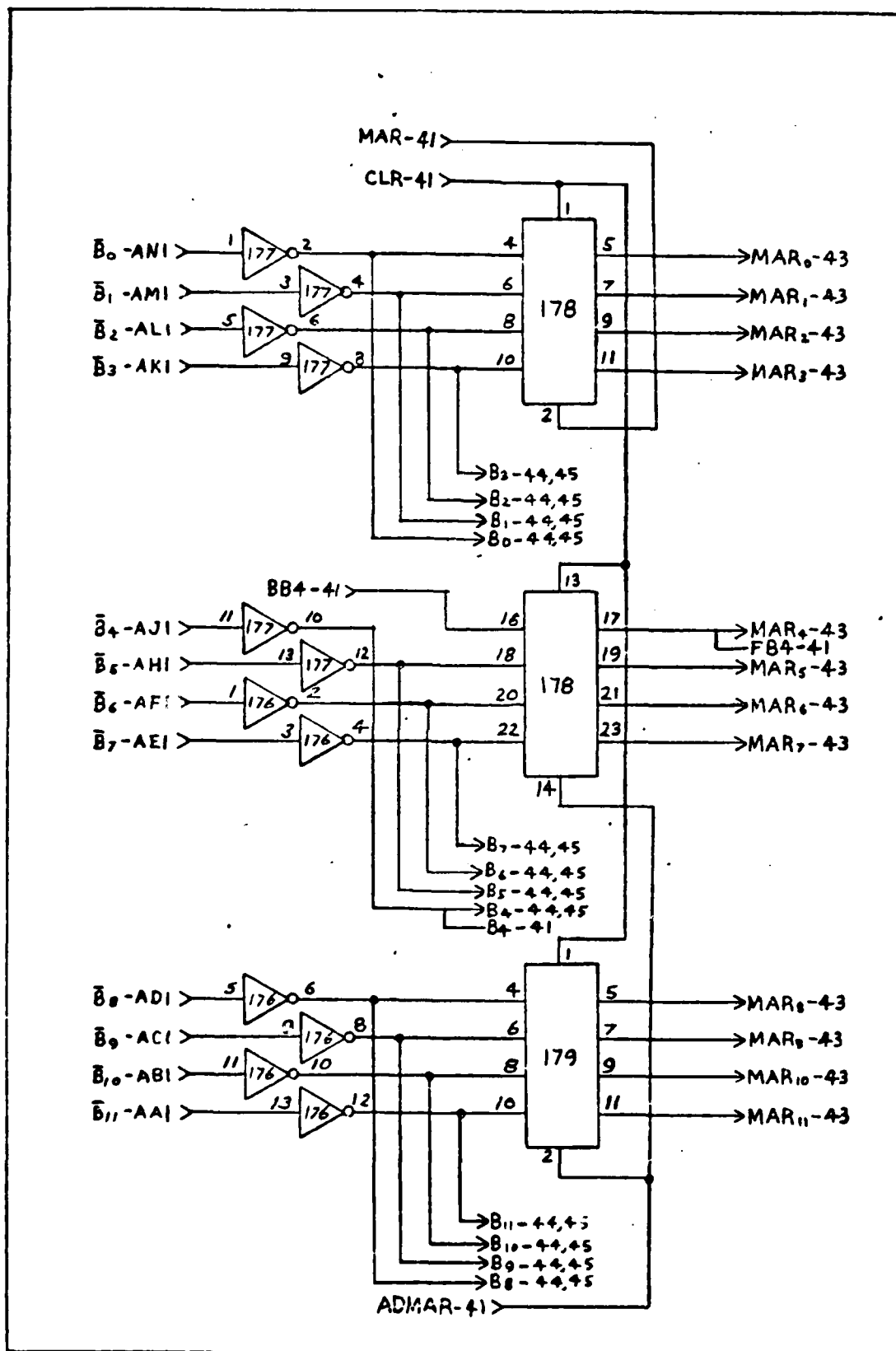


Fig. 42. Memory Address Register

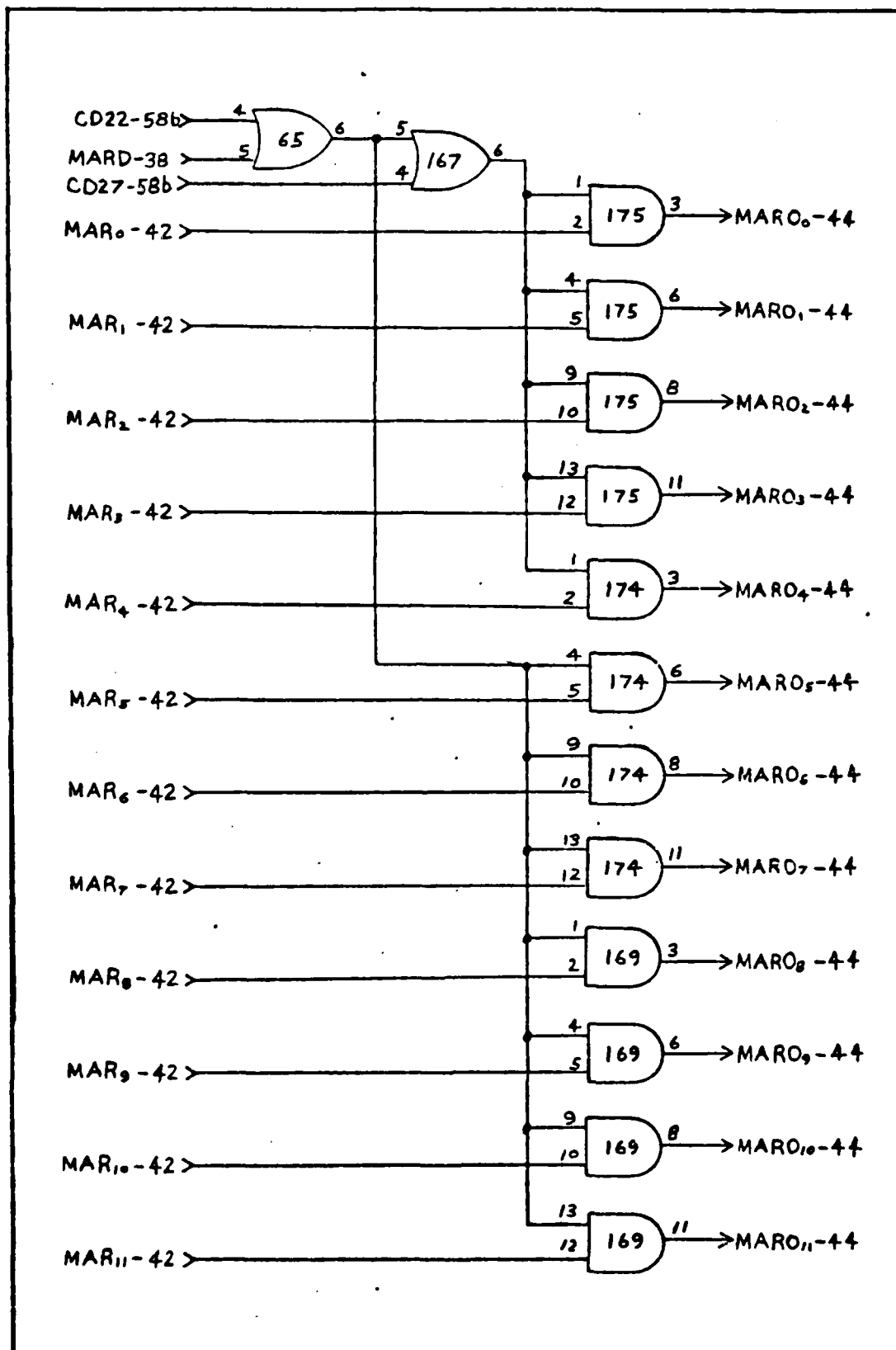
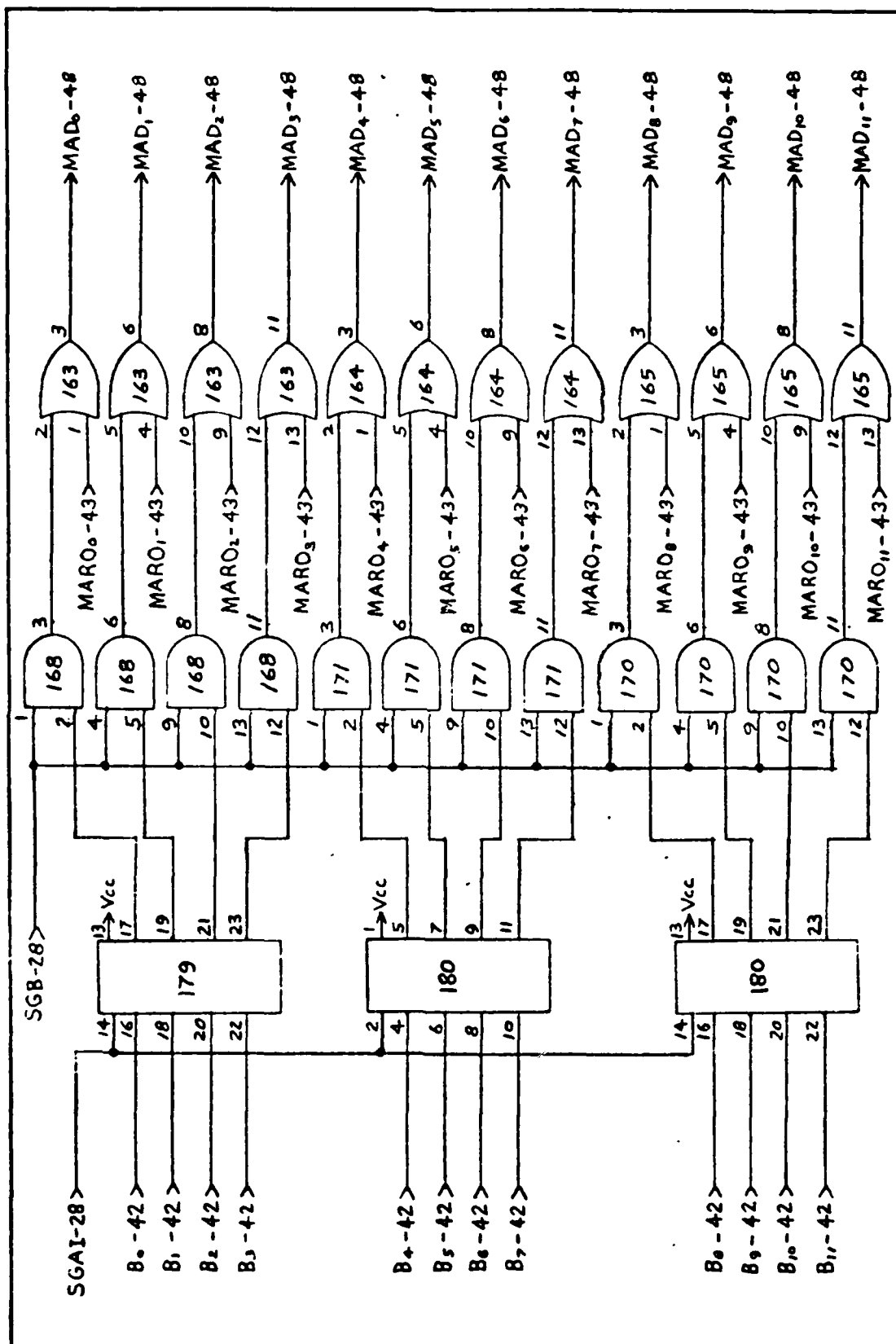


Fig. 43. Memory Address Register Output Switch



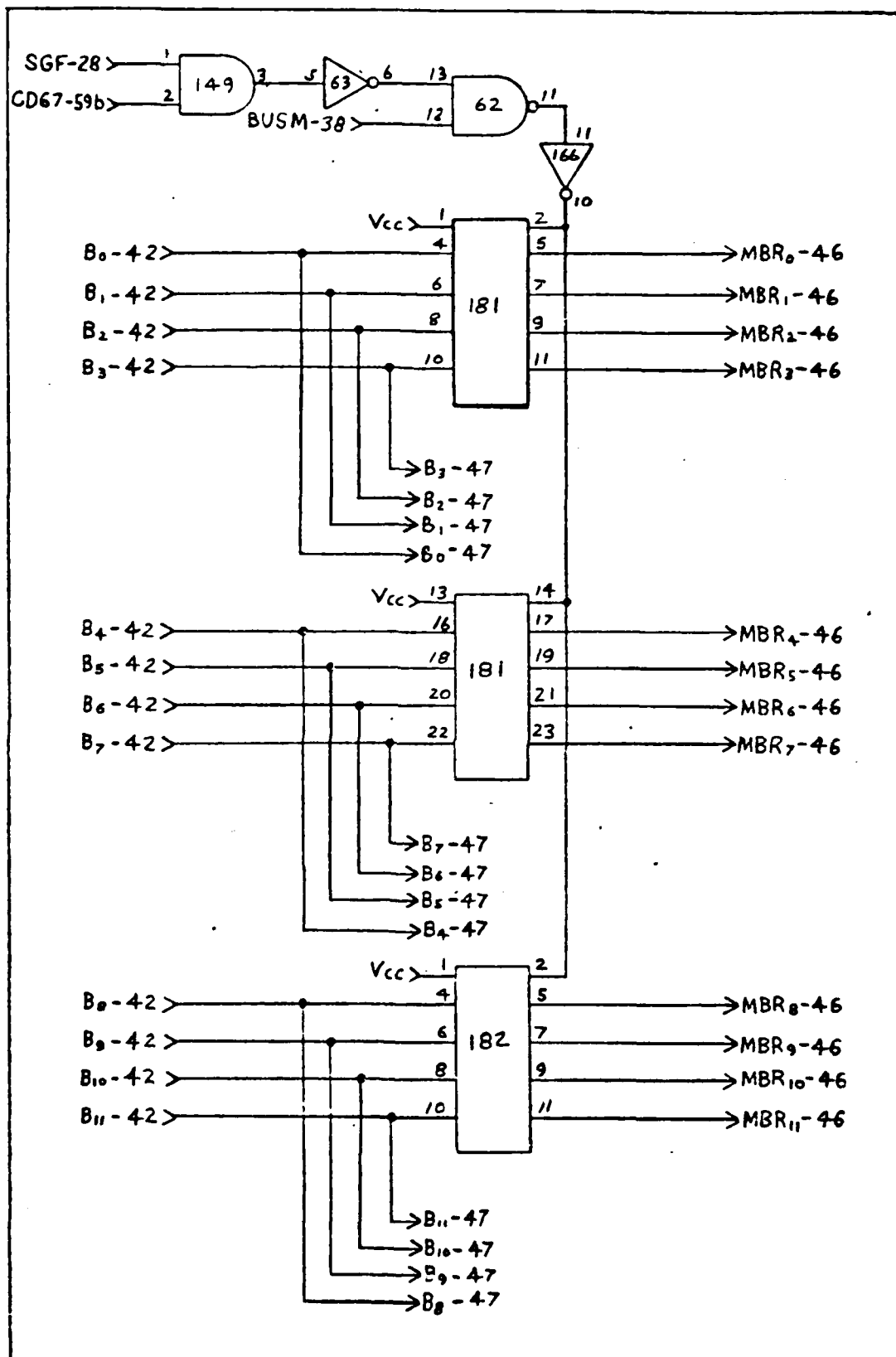


Fig. 45. Memory Buffer Register

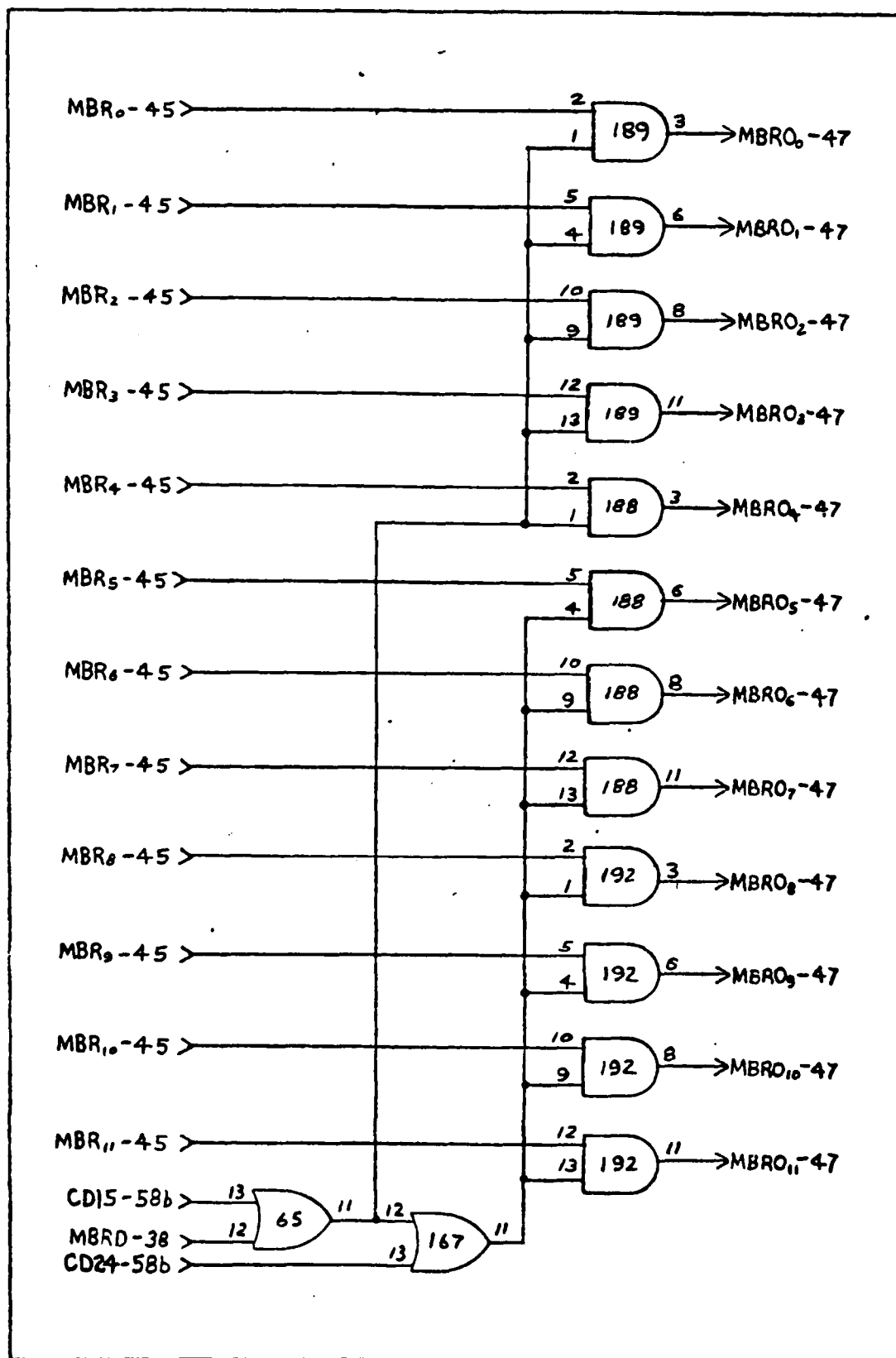


Fig. 46. Memory Buffer Register Output Switch

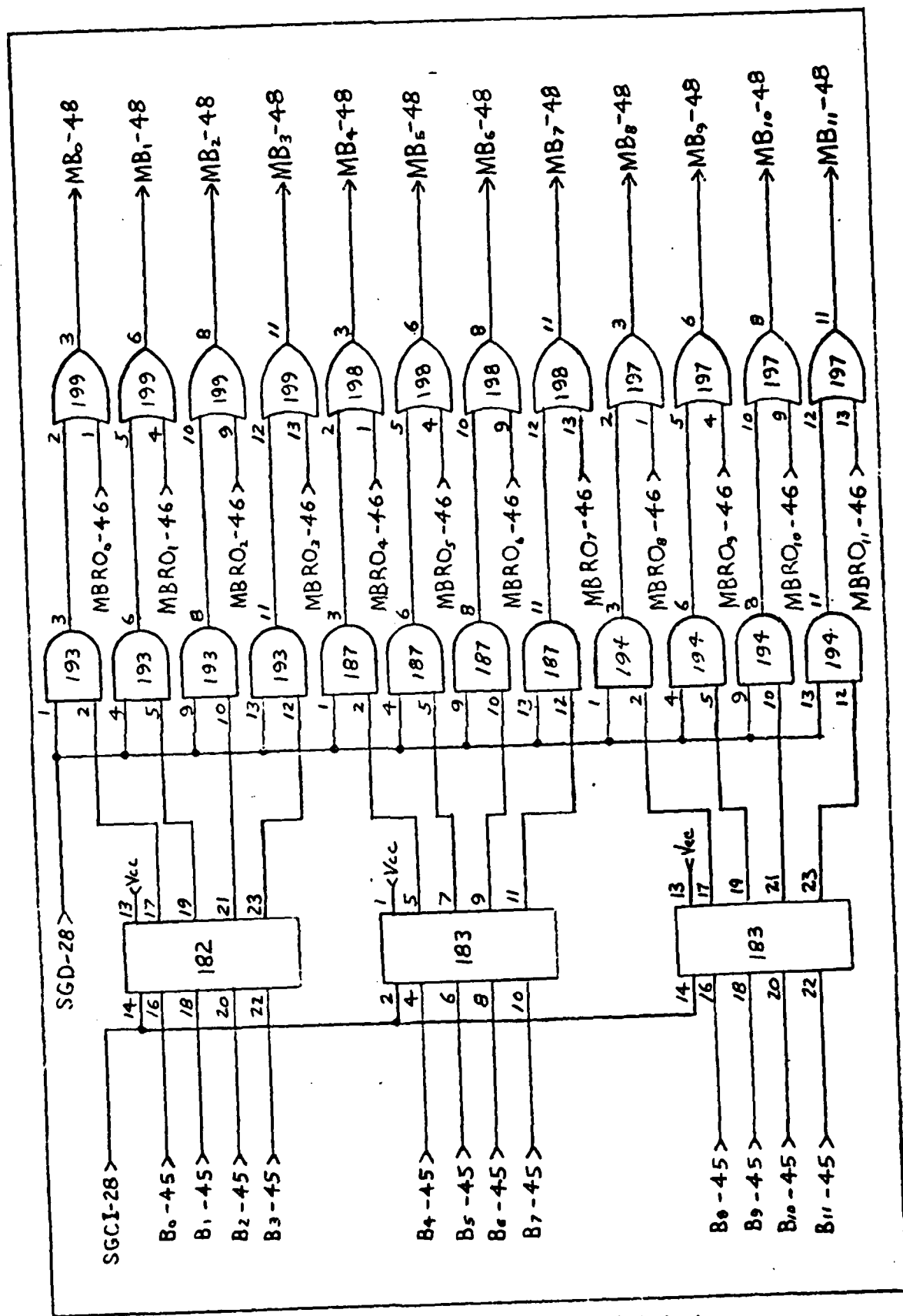


Fig. 47. MBR and MMBR Output Control Switch

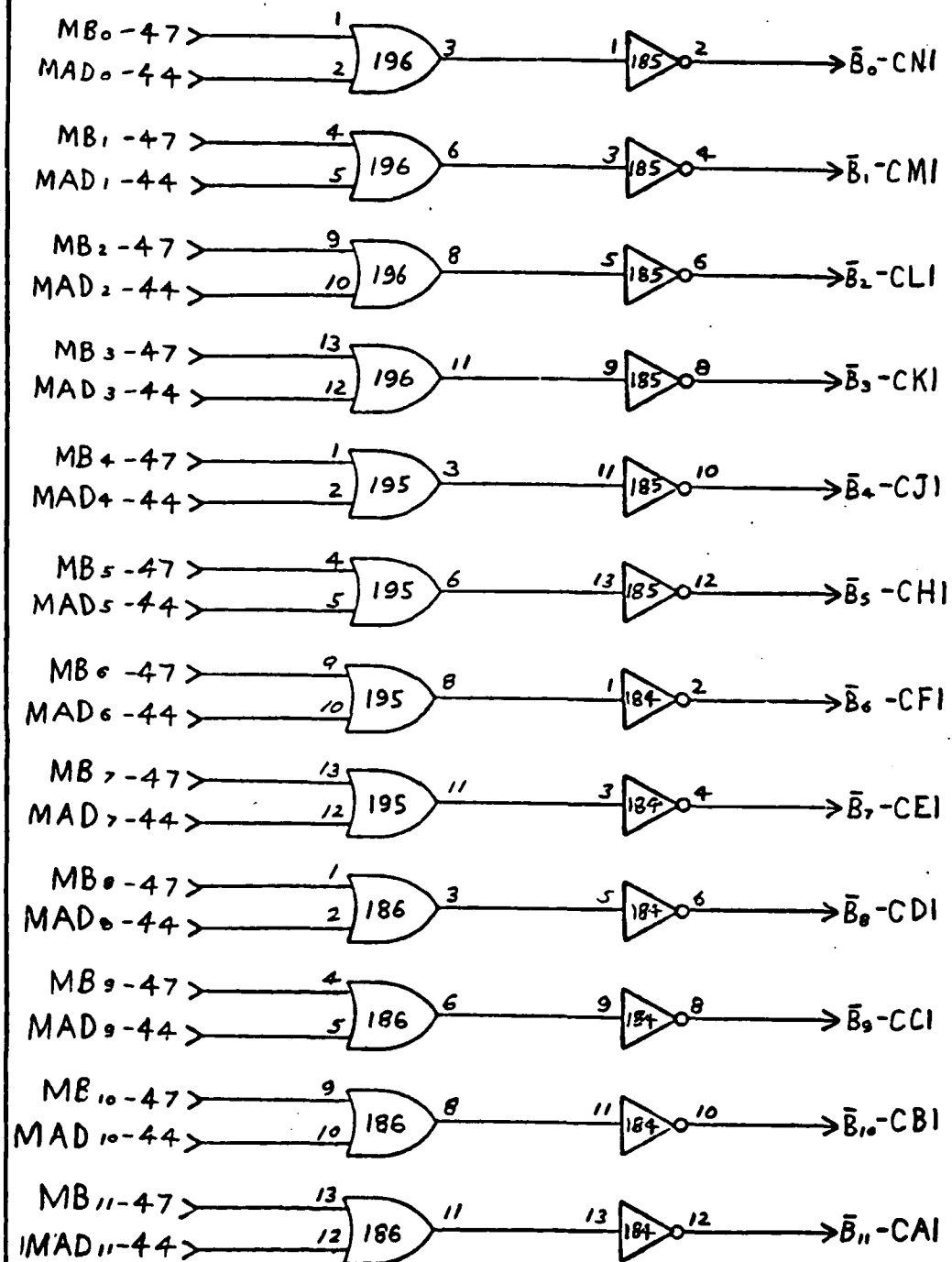


Fig. 48. MAR, MMAR, MBR, and MMBR Bus Control Switch

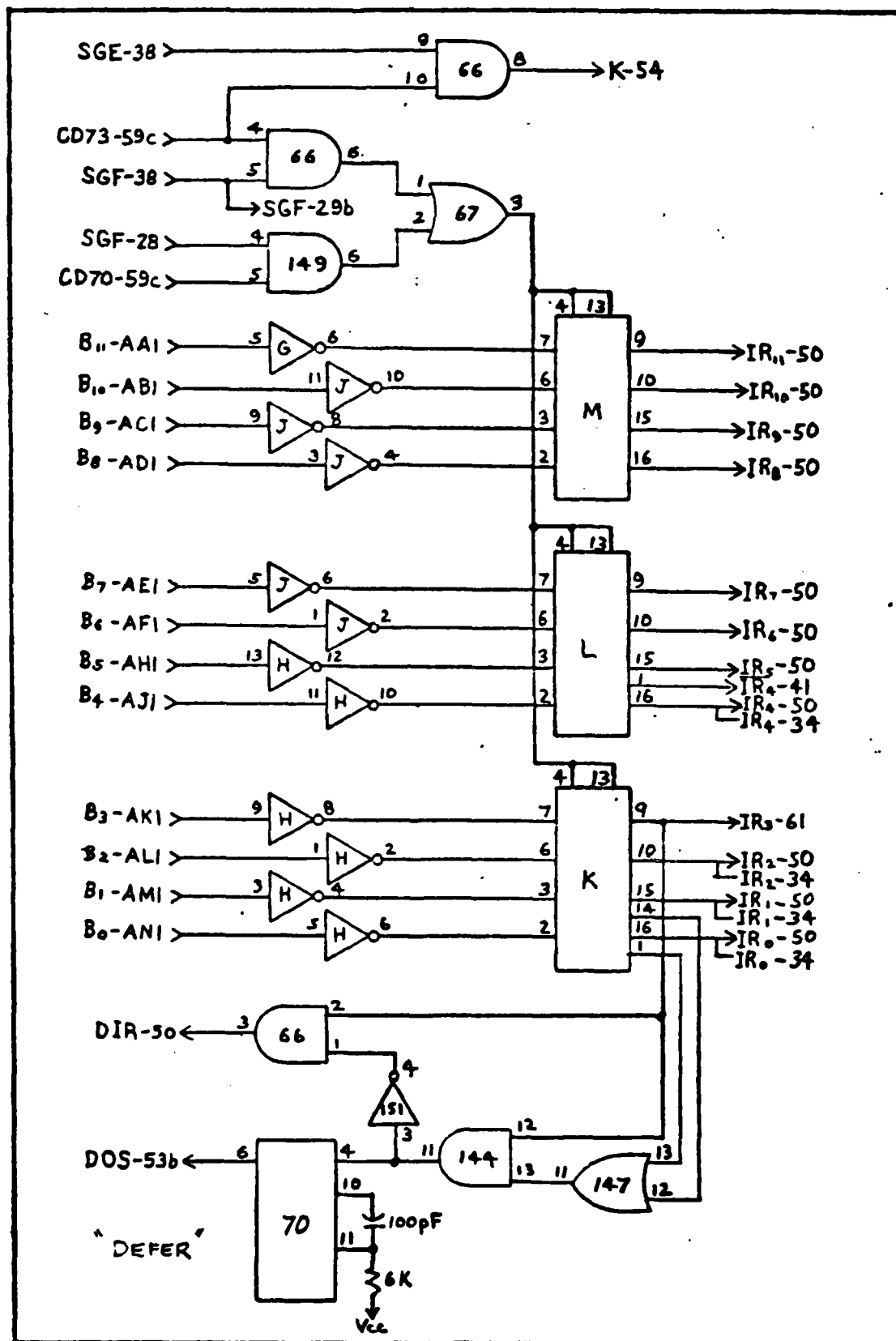


Fig. 49. Instruction Register

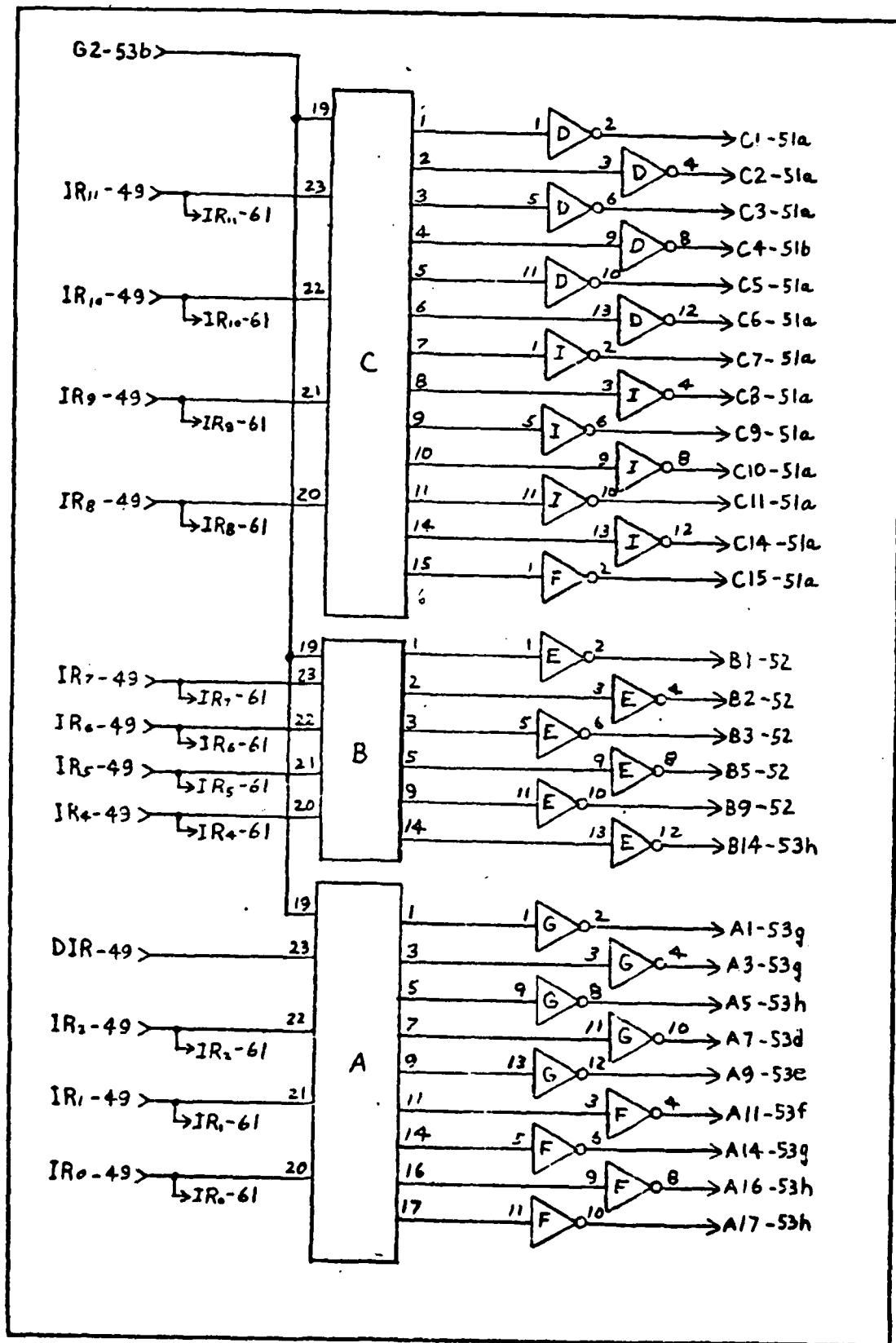


Fig. 50. Instruction Register Decoder

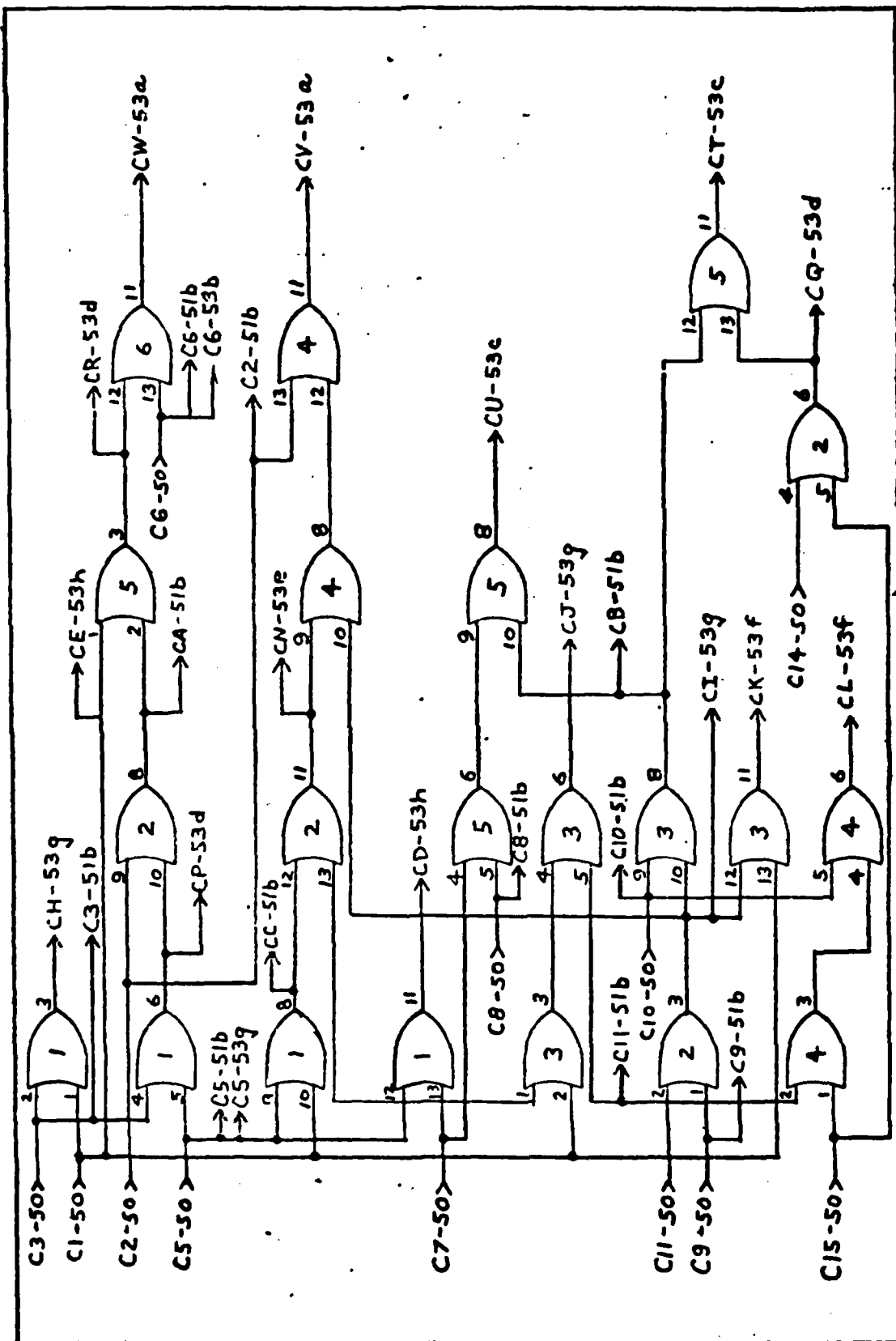


Fig. 51a. C-Logic

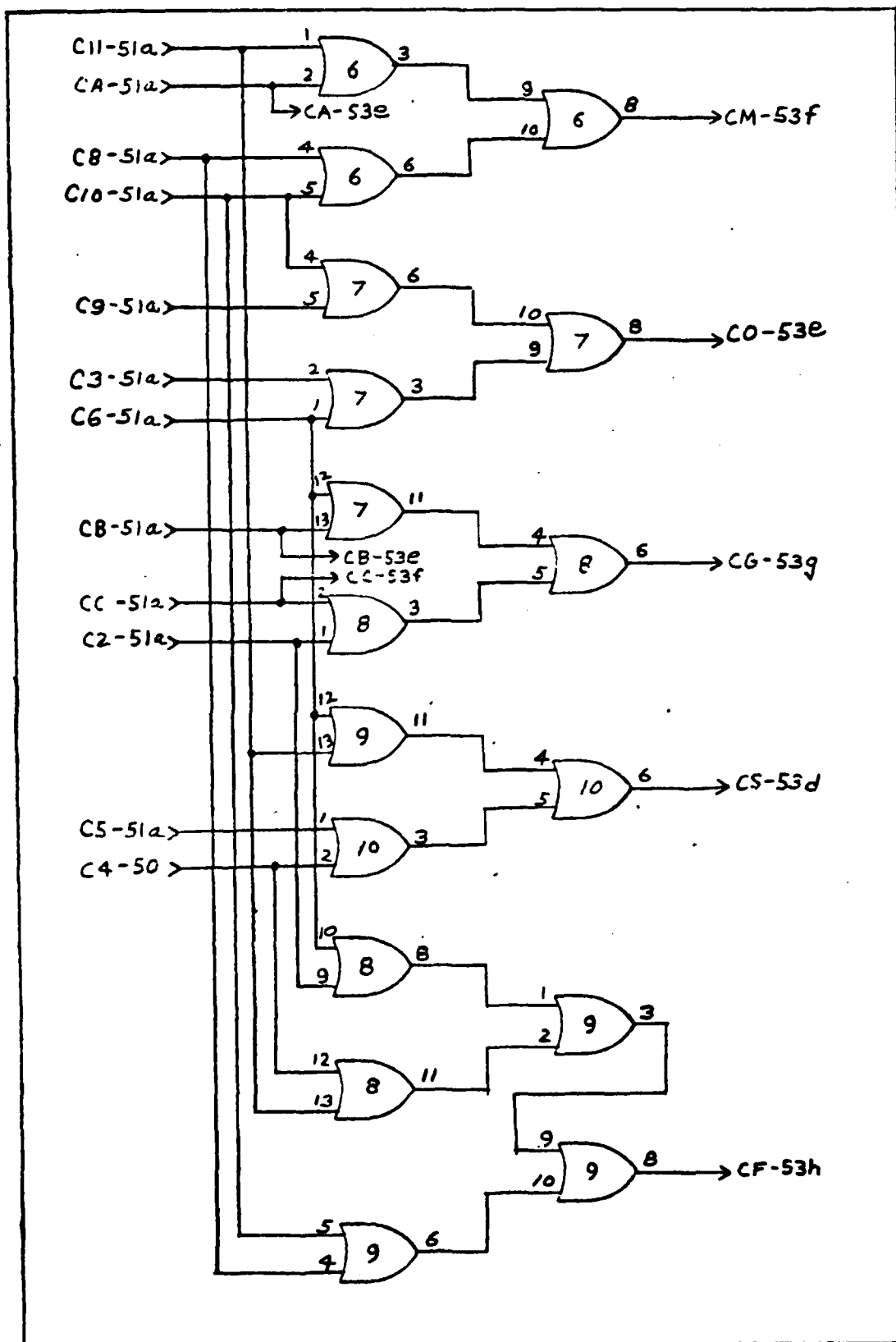


Fig. 51b. C-Logic

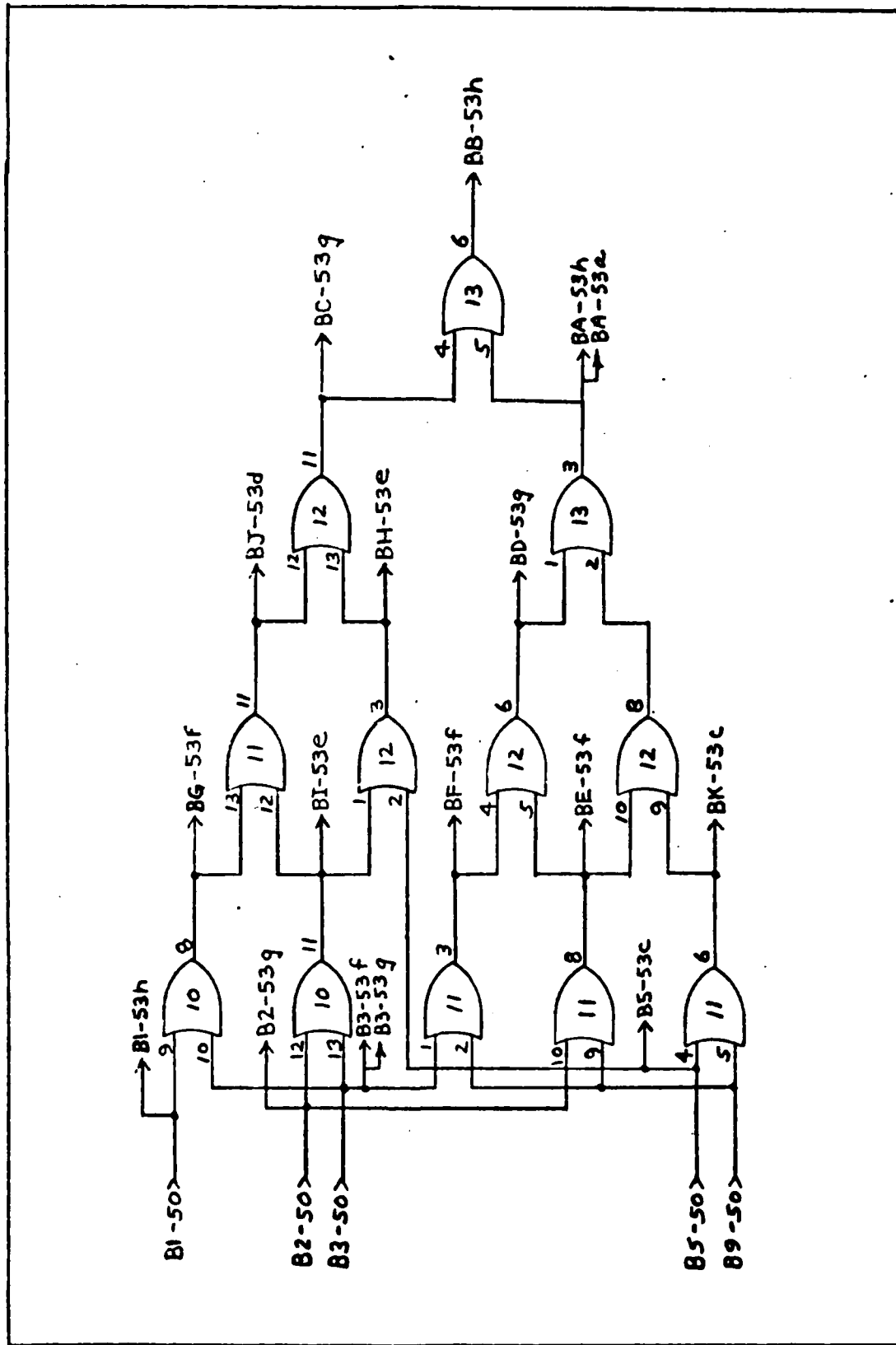


Fig. 52. B-Logic

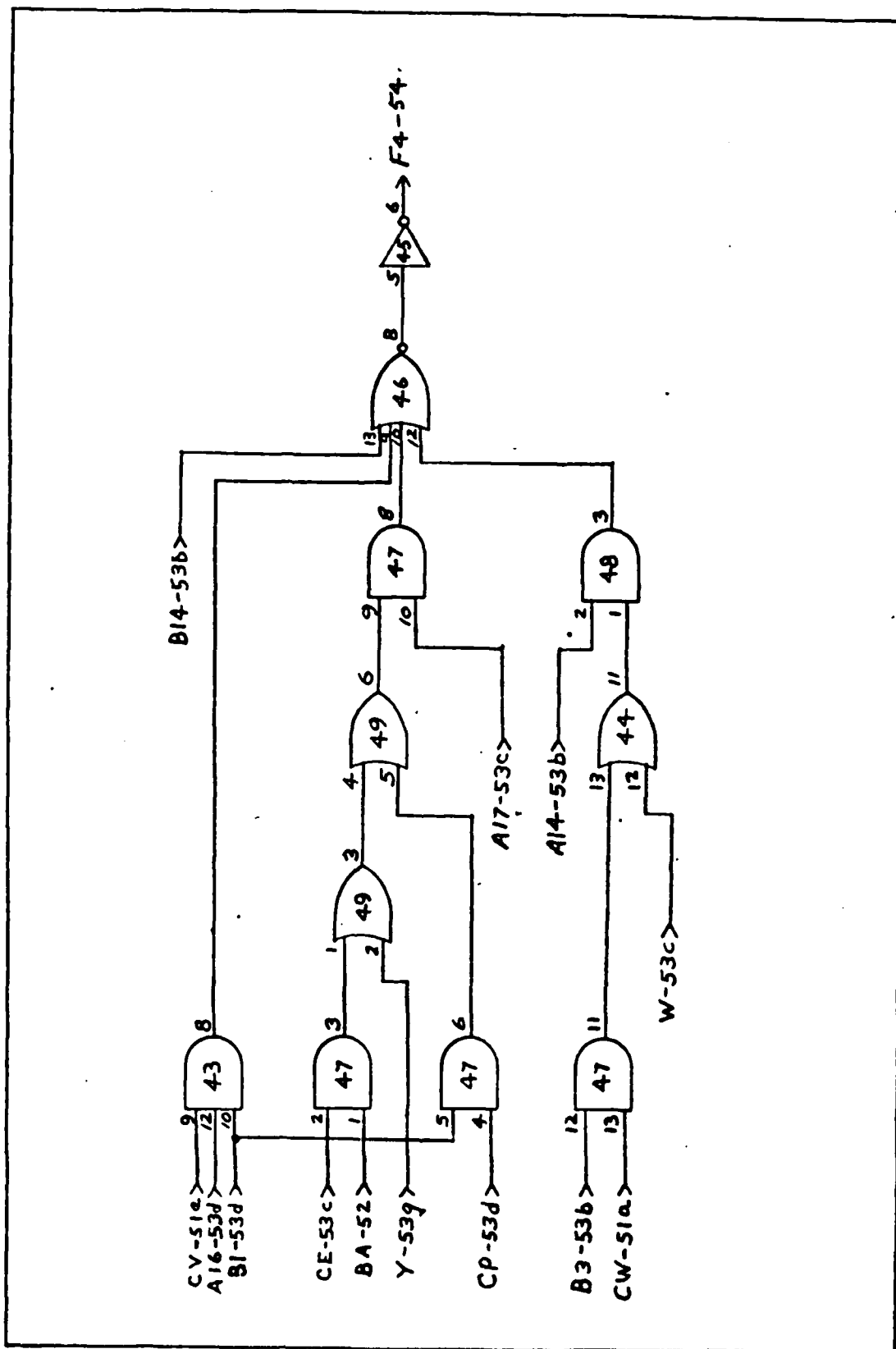


Fig. 53a. Microprogram Program Counter Input Signal (F4)

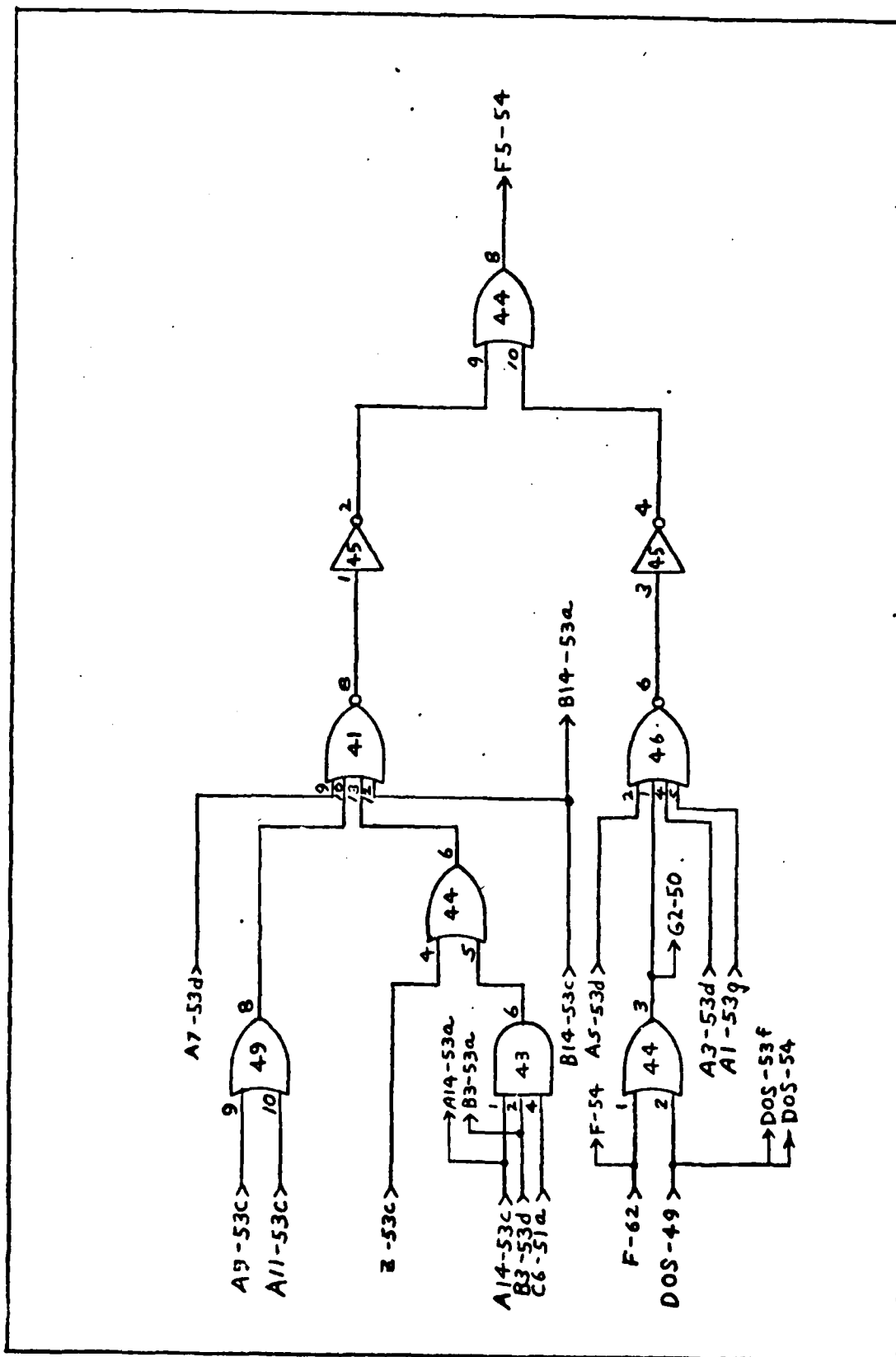


Fig. 53b. Microprogram Program Counter Input Signal (F5)

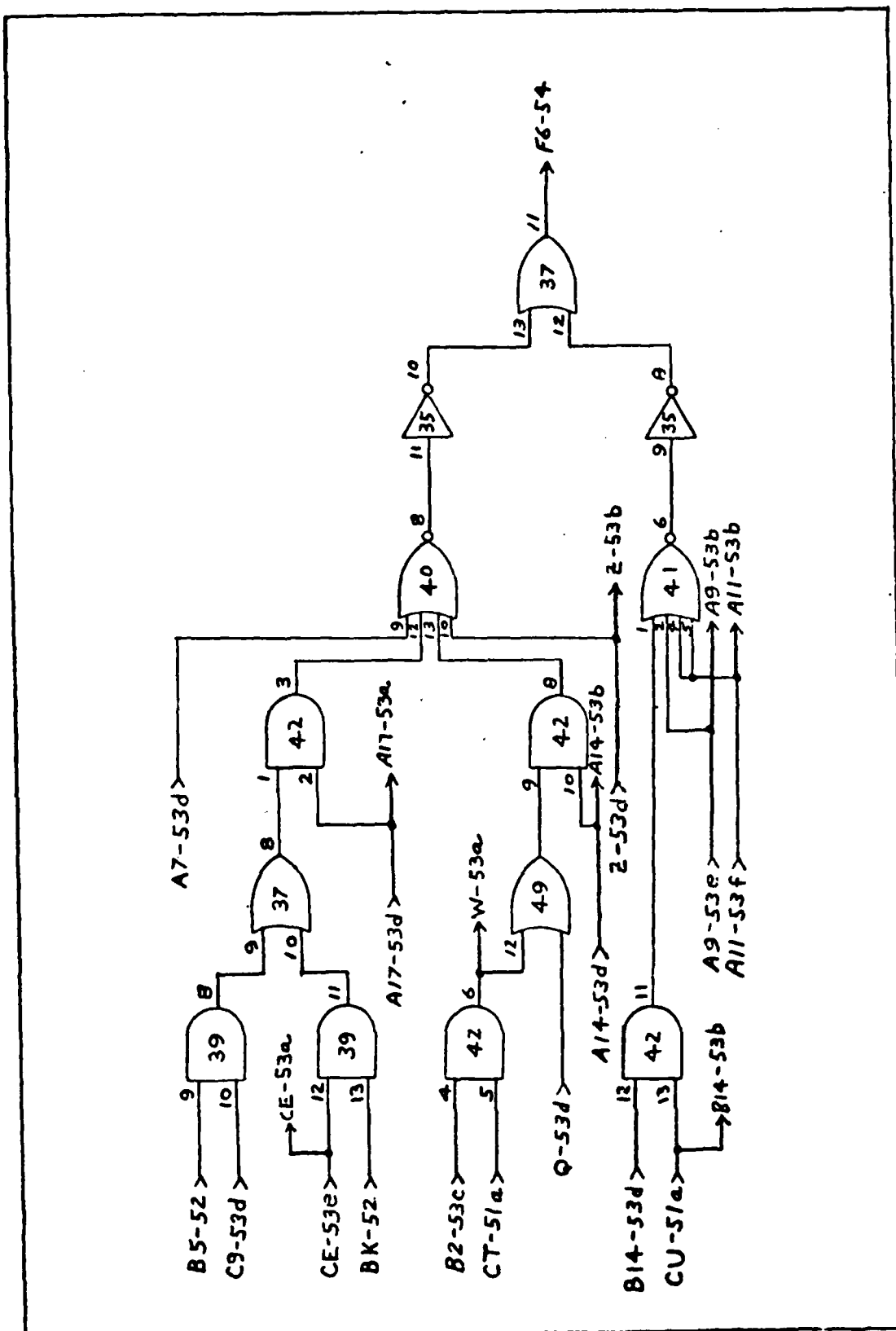


Fig. 53c. Microprogram Program Counter Input Signal (F6)

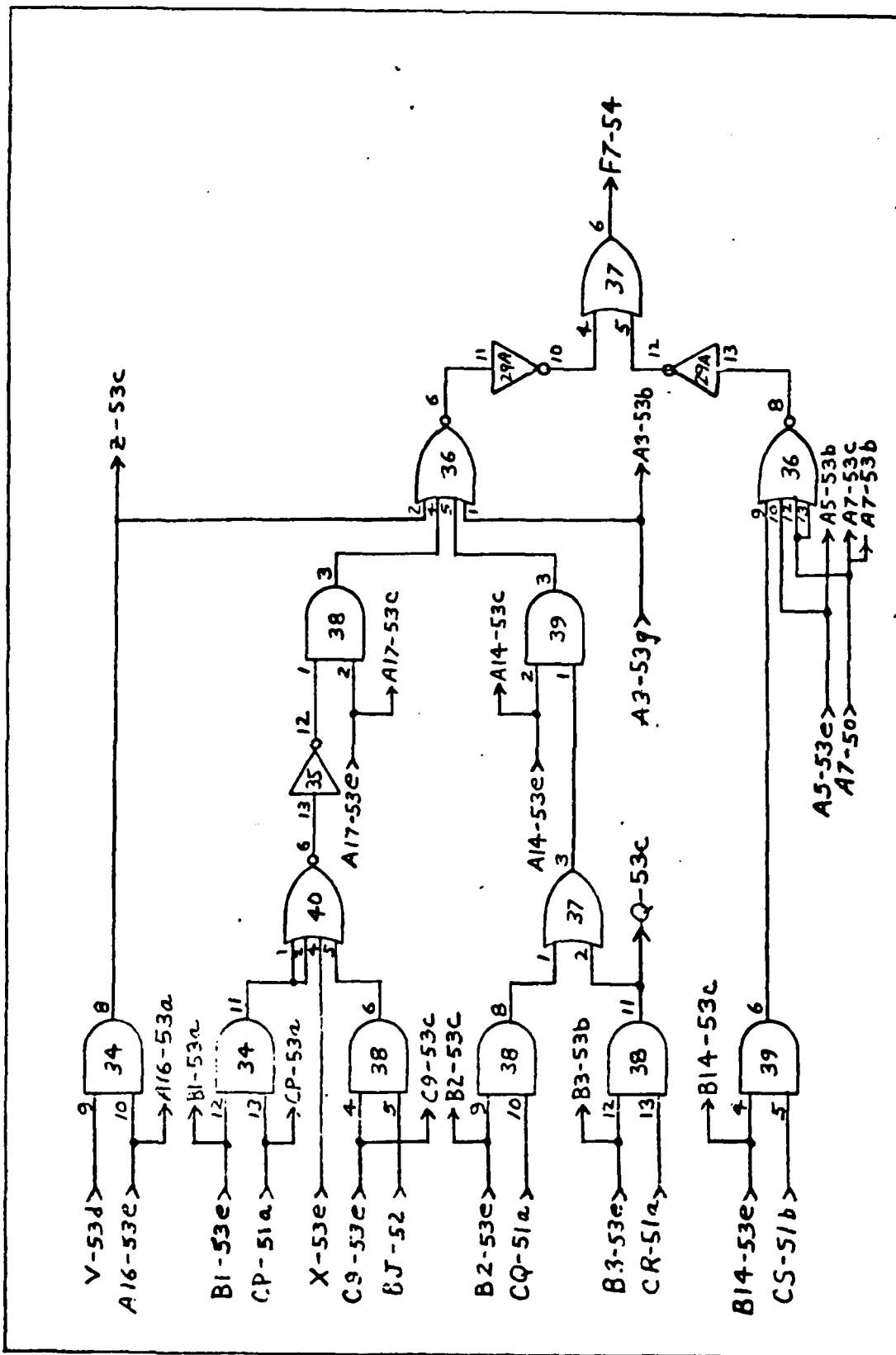


Fig. 53d. Microprogram Program Counter Input Signal (F7)

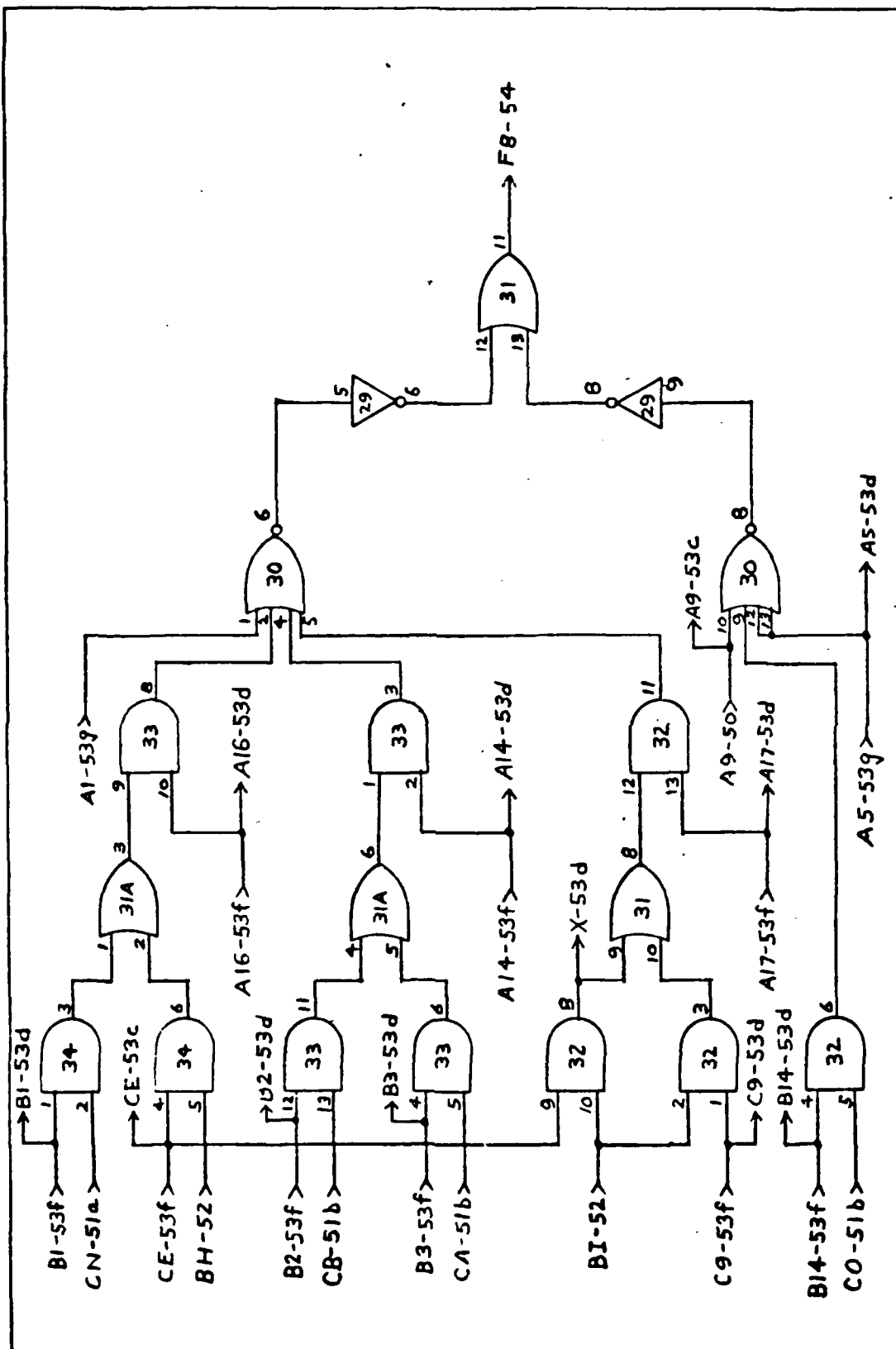


Fig. 53e. Microprogram Program Counter Input Signal (F8)

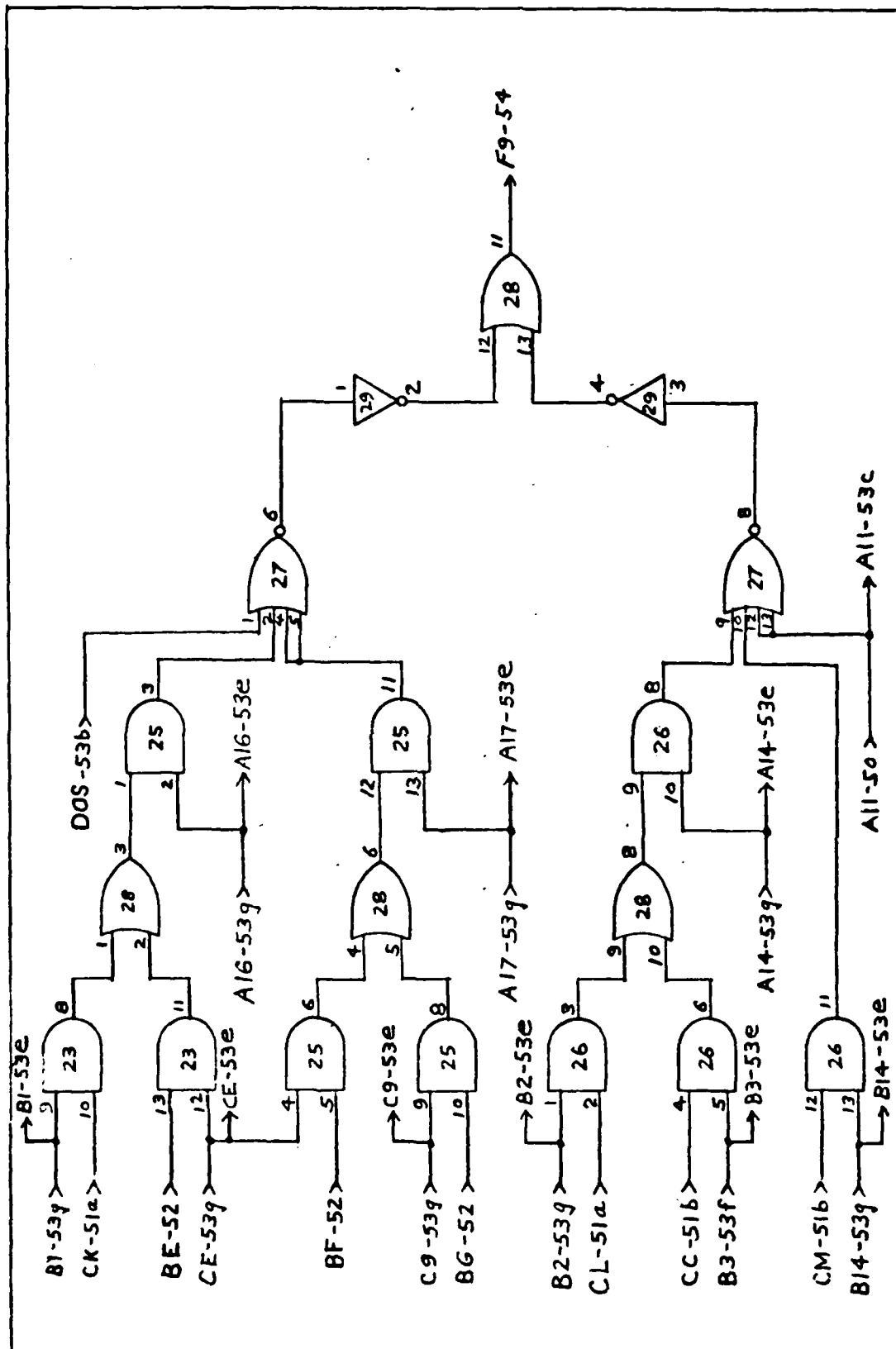


Fig. 53f. Microprogram Program Counter Input Signal (F9)

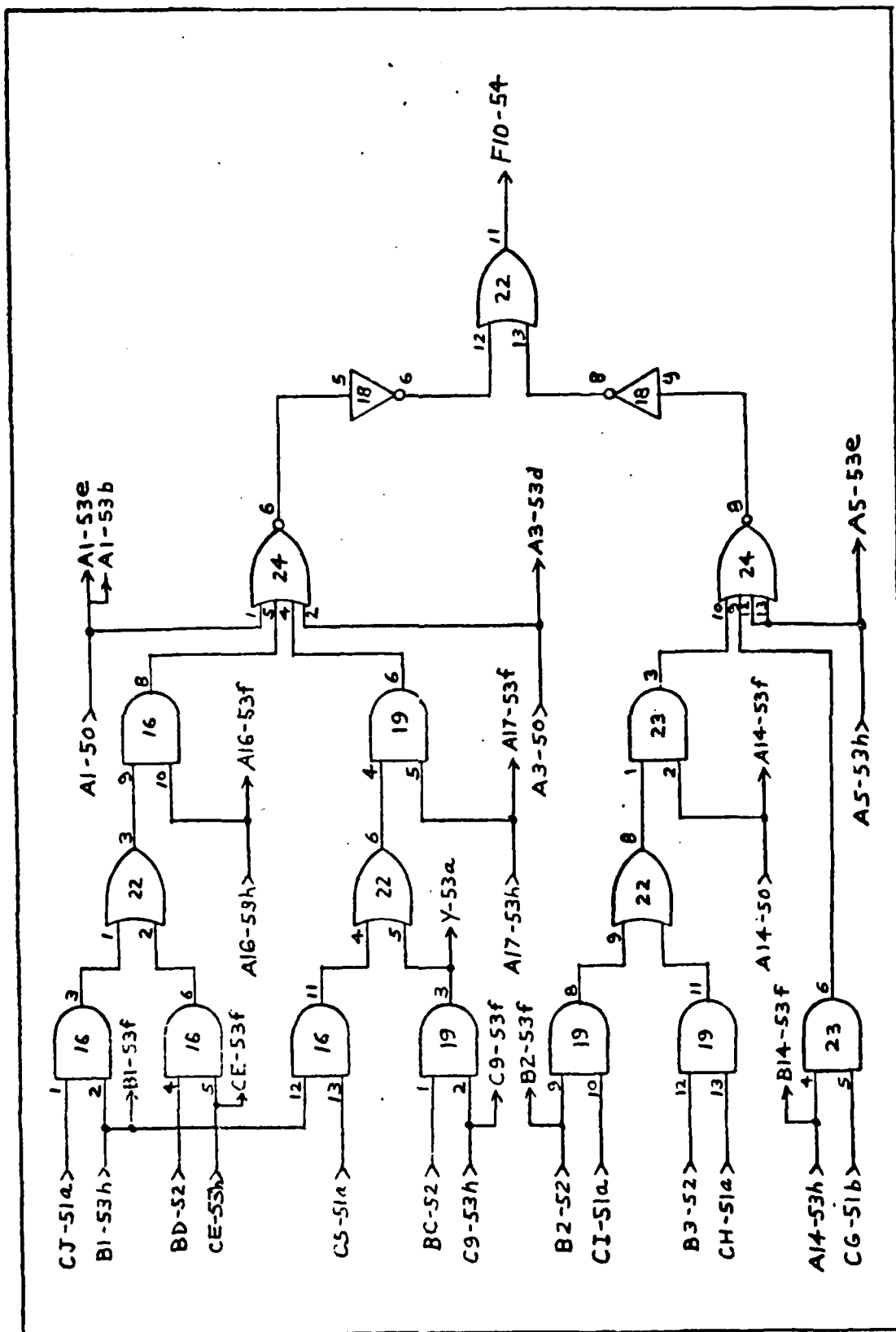


Fig. 53g. Microprogram Program Counter Input Signal (F10)



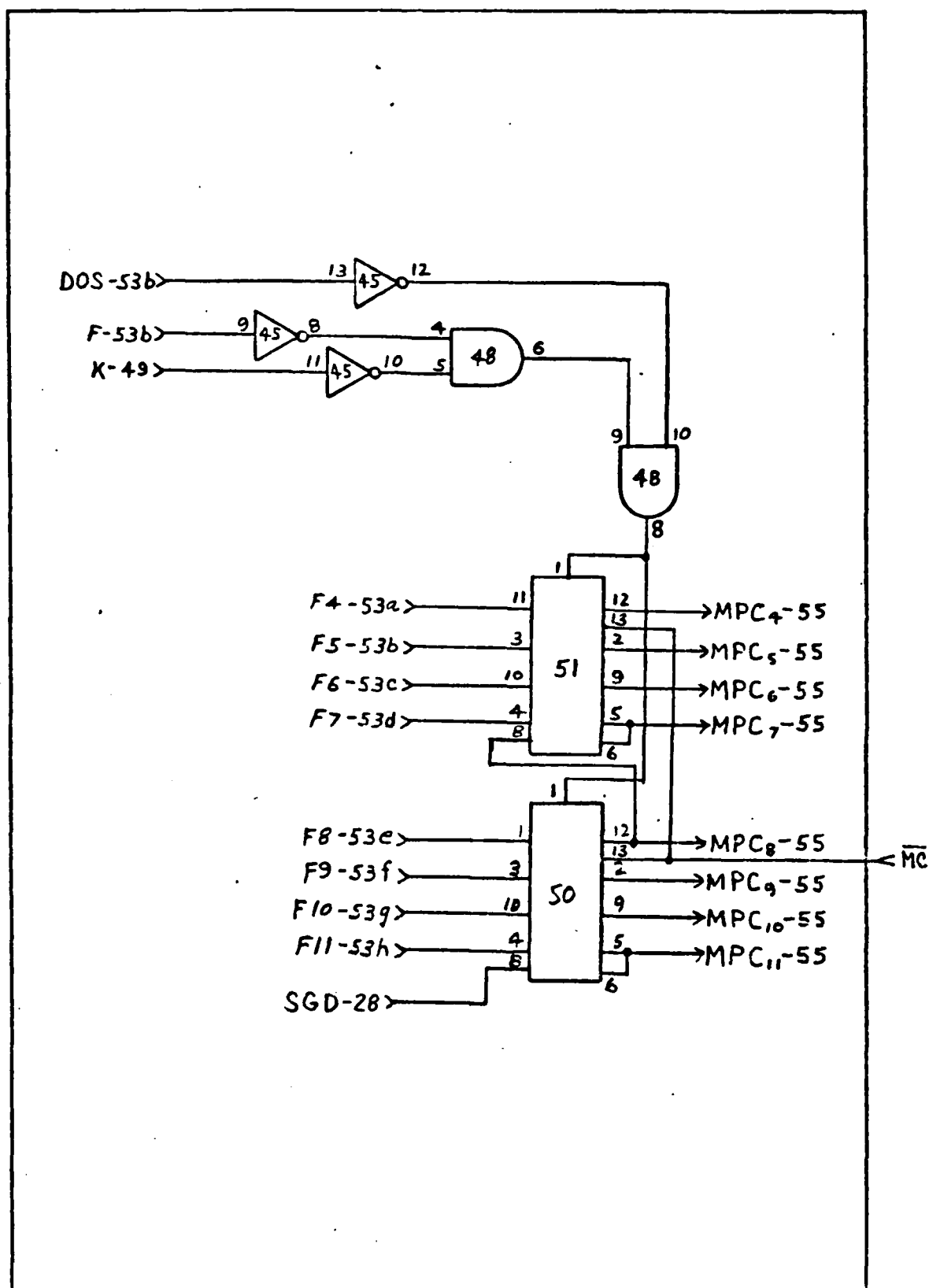


Fig. 54. Microprogram Program Counter

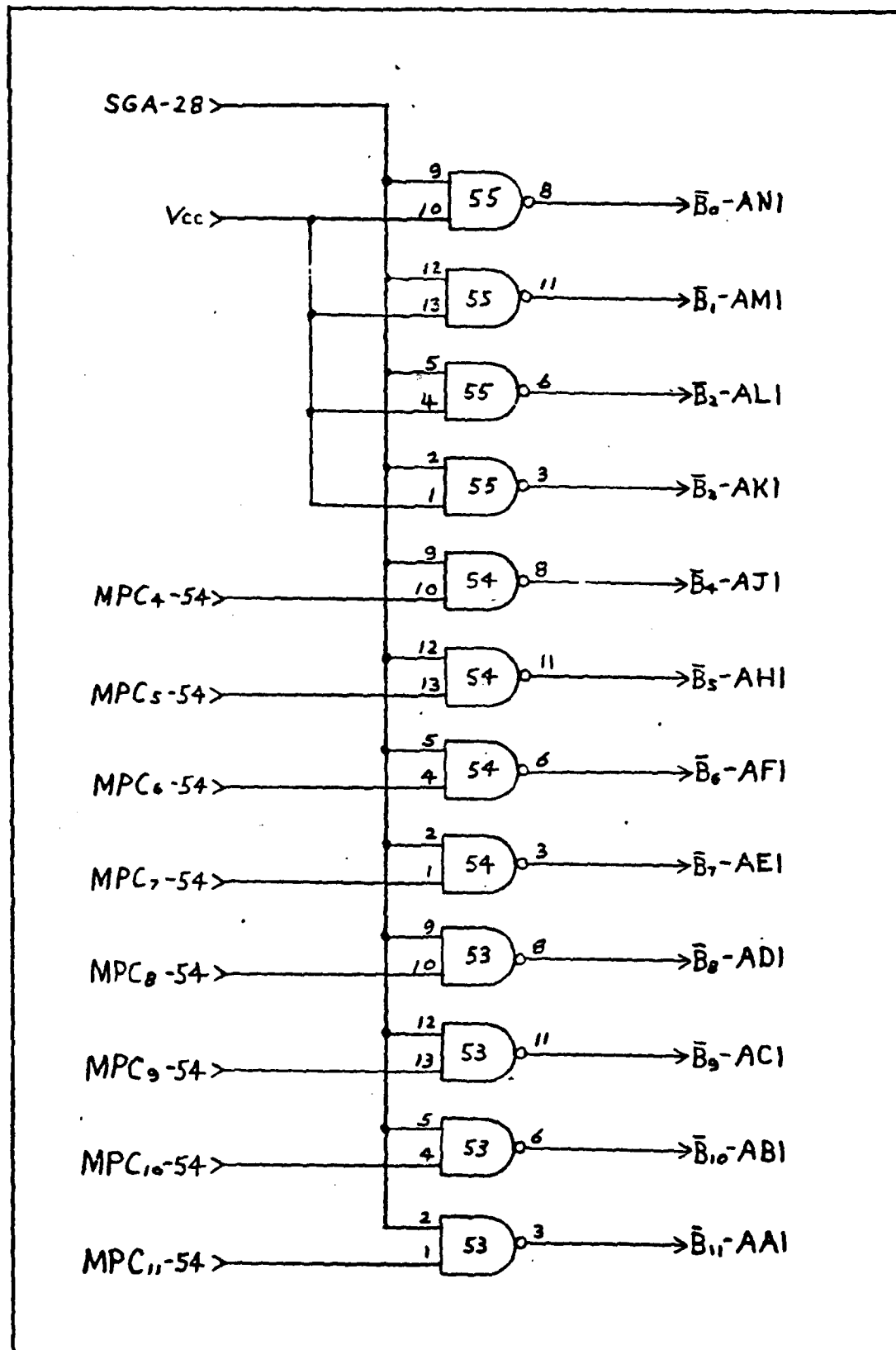


Fig. 55. Microprogram Program Counter Bus Control Switch

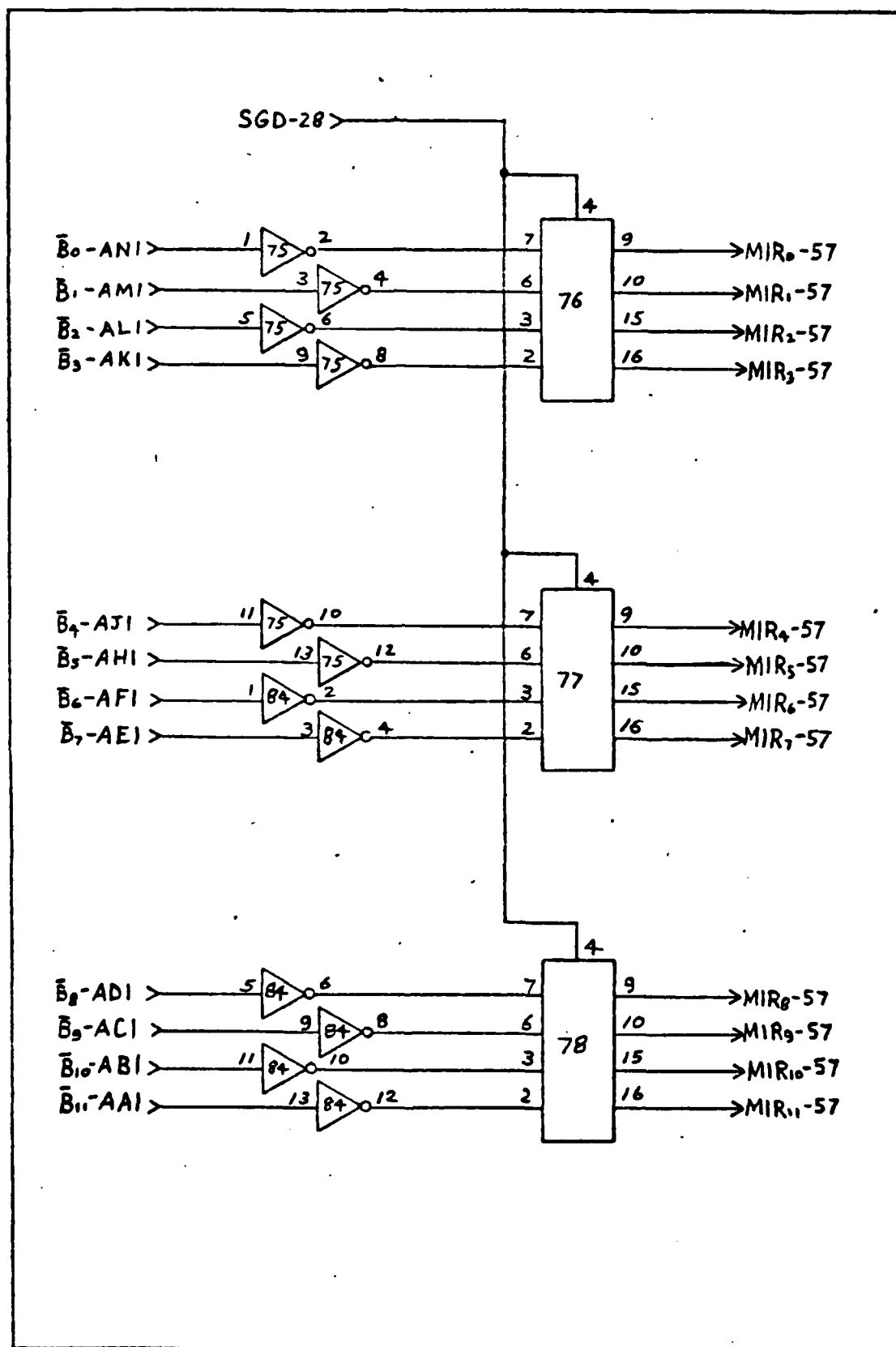


Fig. 56. Microprogram Instruction Register

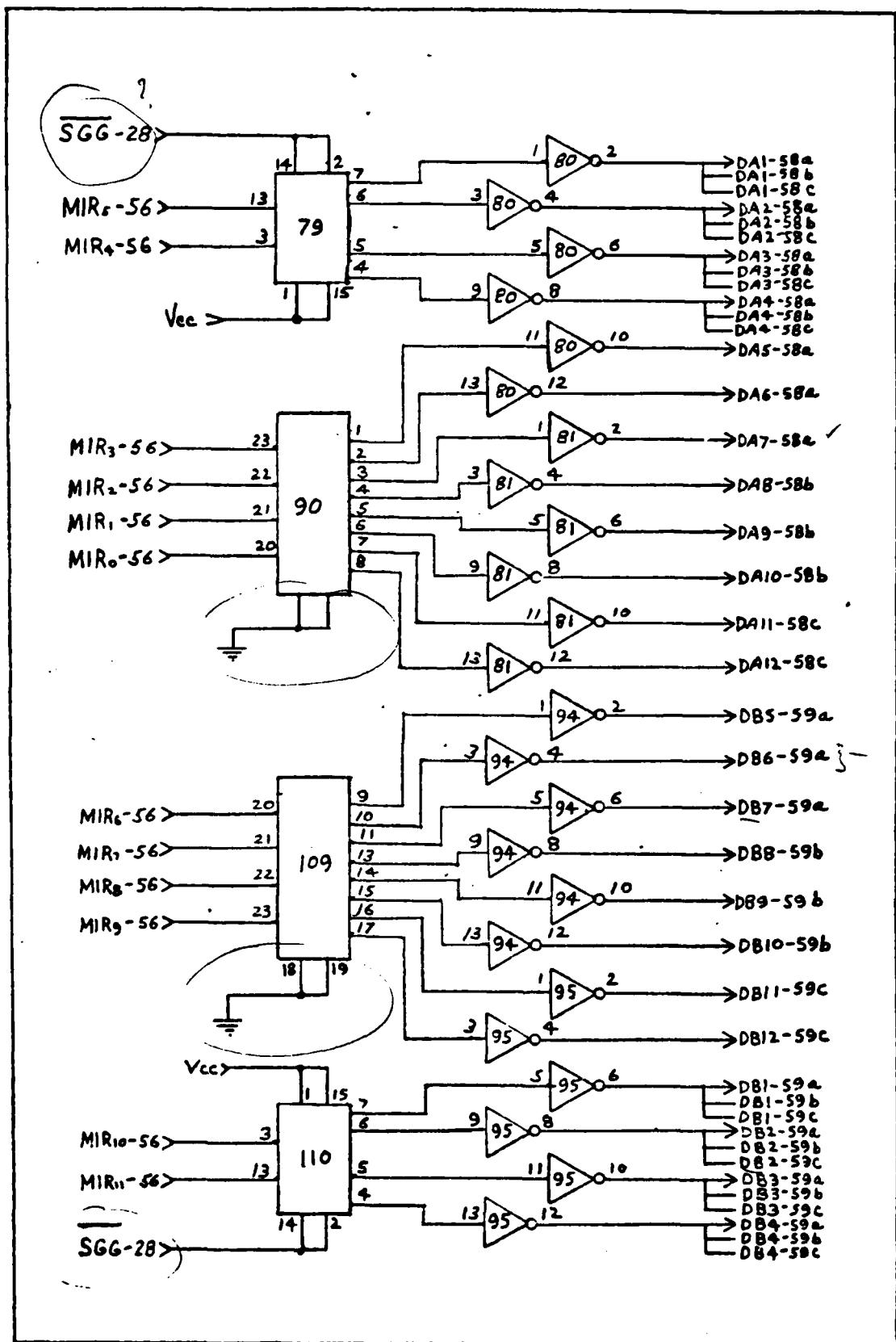


Fig. 57. Microprogram Instruction Register Decoder

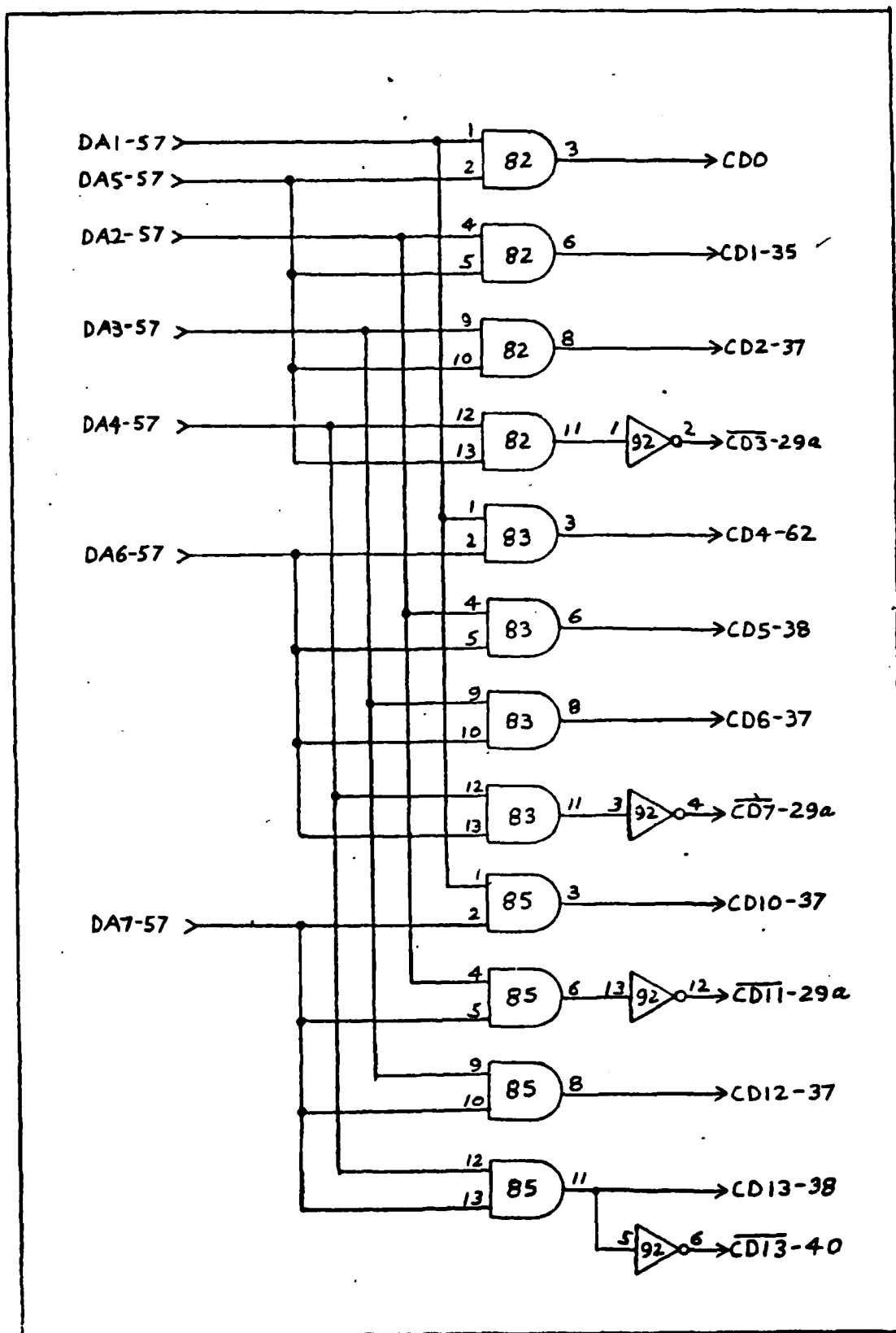


Fig. 58a. Control Switch For Field A (Codes 0 - 13)

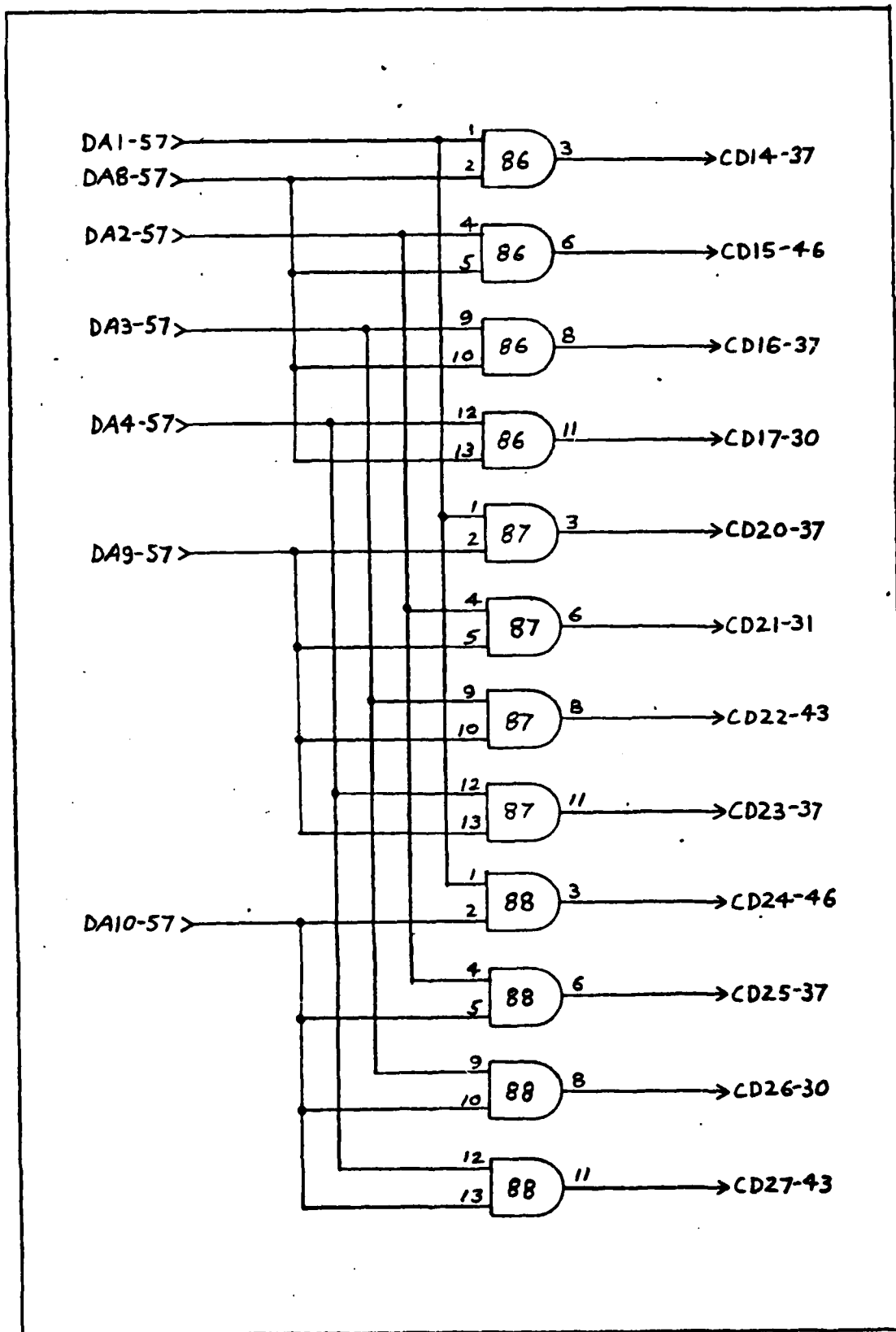


Fig. 58b. Control Switch For Field A (Codes 14 - 27)

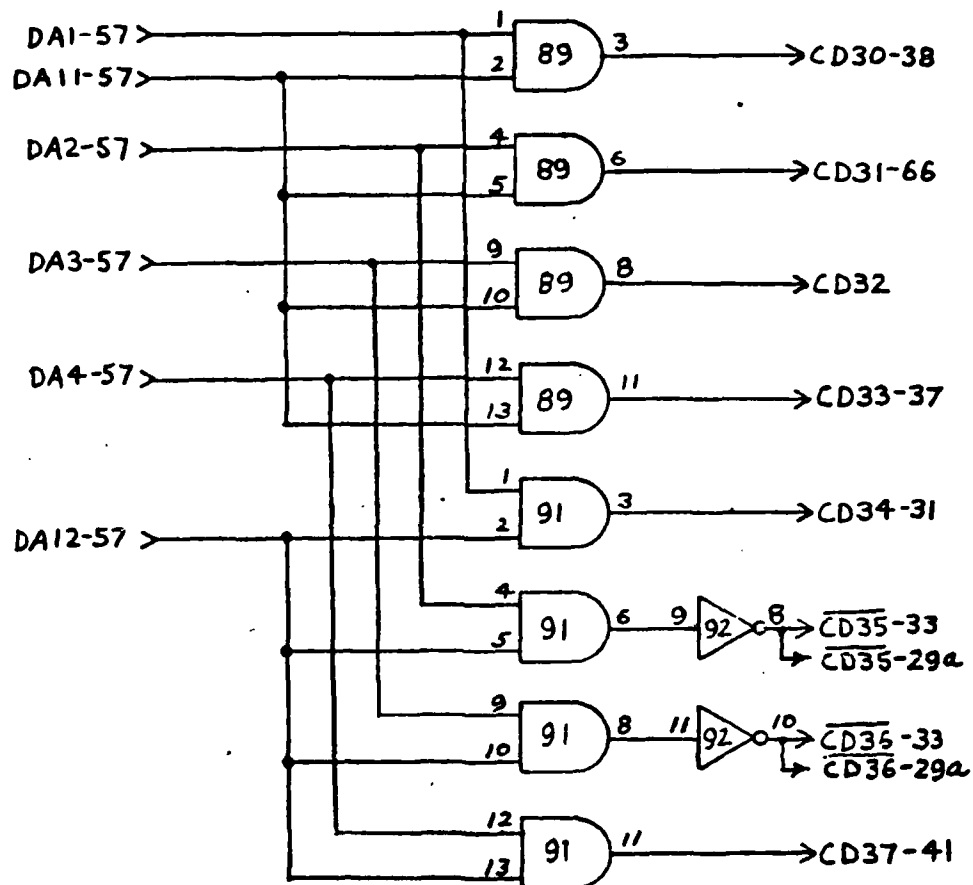


Fig. 58c. Control Switch For Field A (Codes 30 - 37)

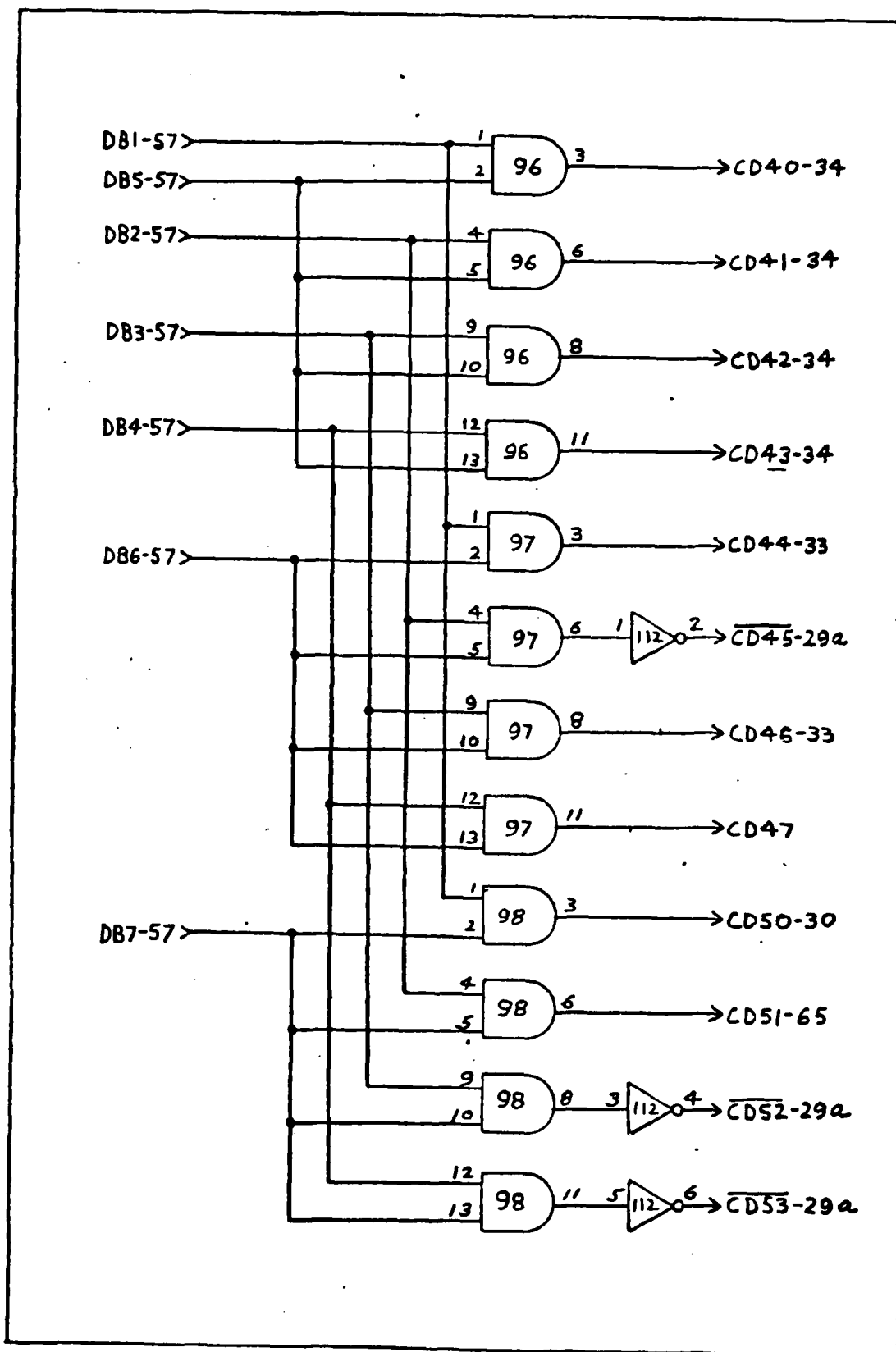


Fig. 59a. Control Switch For Field B (Codes 40 - 53)

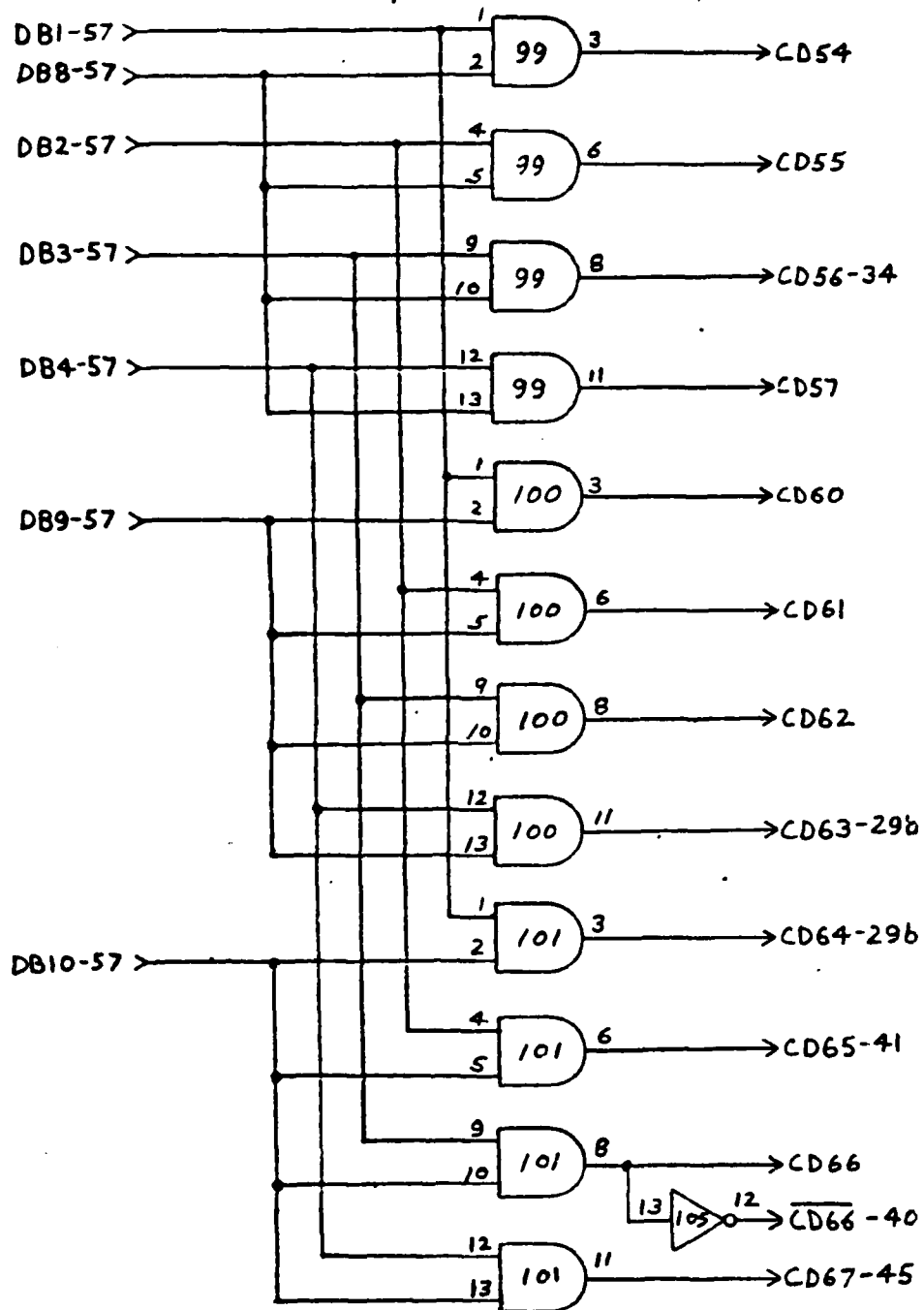
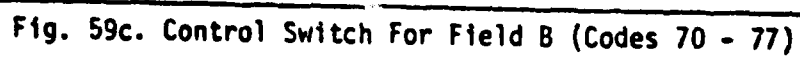


Fig. 59b. Control Switch For Field B (Codes 54 - 67)



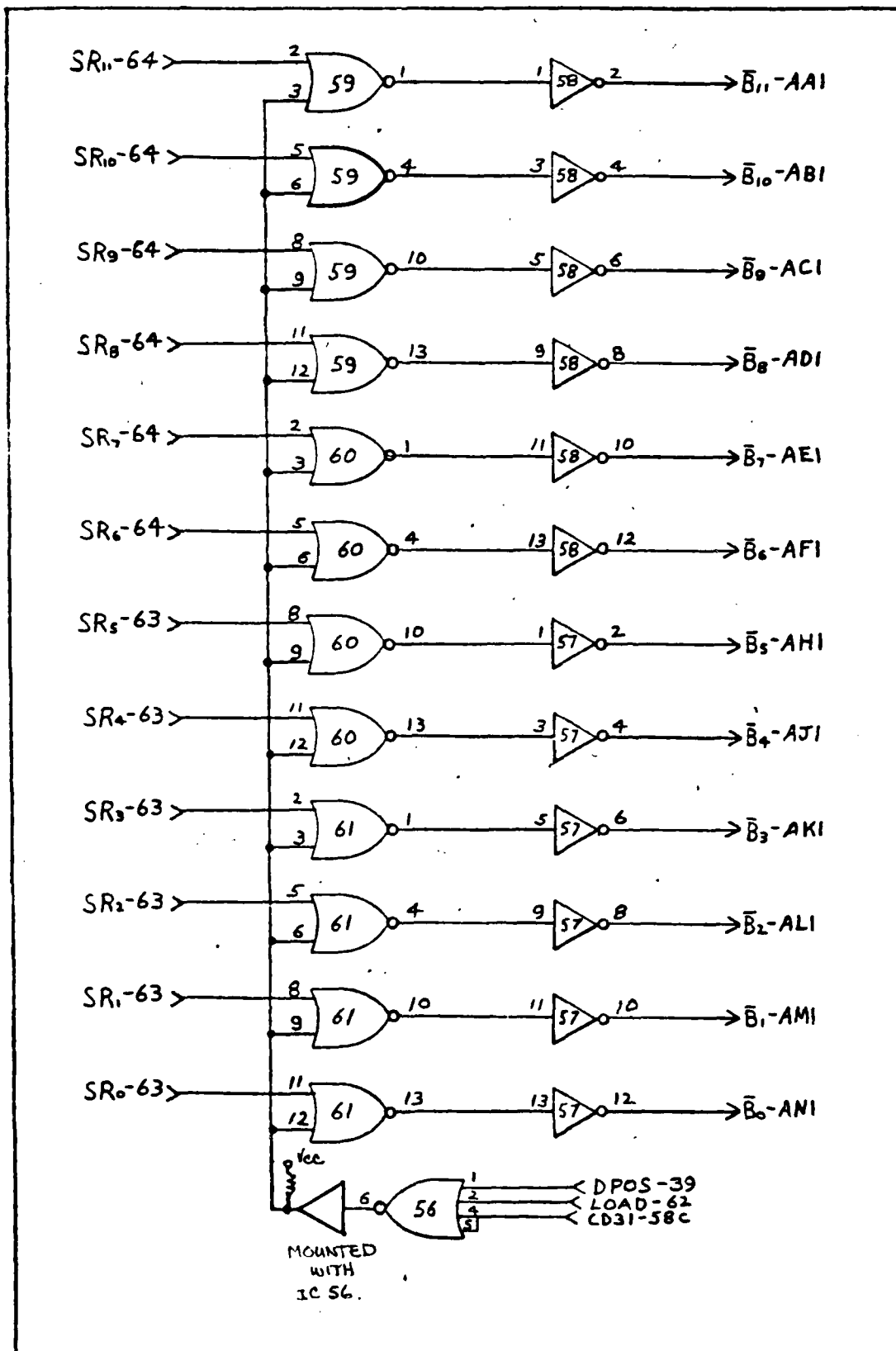


Fig. 66. Switch Register

## Appendix D

### Backplane Pin-to-Pin Wiring

# Module 1. Instruction Register and Instruction Register Decoder

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
AA1	$\overline{B}_{11}/G(5)$		
AB1	$\overline{B}_{10}/J(11)$		
AC1	$\overline{B}_9/J(9)$		
AD1	$\overline{B}_8/J(3)$		
AE1	$\overline{B}_7/J(5)$		
AF1	$\overline{B}_6/J(1)$		
AH1	$\overline{B}_5/H(13)$		
AJ1	$\overline{B}_4/H(11)$		
AK1	$\overline{B}_3/H(9)$		
AL1	$\overline{B}_2/H(1)$		
AM1	$\overline{B}_1/H(3)$		
AN1	$\overline{B}_0/H(5)$		
AP1	$IR_2/K(10)$	4/BK2	$IR_2/64(12)$
AR1	$C3/1(4)$	2/AR1	$C3/7(2)$
AS1	$IR_1/K(15)$	4/BJ2	$IR_1/63(11)$
AU1	$\overline{IR}_4/L(1)$	8/DN2	$\overline{IR}_4/200(10)$
AV1	$IR_4/L(16)$	7/BS1	$IR_4/140(2)$
AB2	$C6/D(12)$	2/AB2	$C6/6(13)$
AD2	$C11/2(2)$	2/AD2	$C11/6(1),9(13)$
AE2	$C10/3(9)$	2/AE2	$C10/6(5)$
AF2	$C8/5(5)$	2/AF2	$C8/6(4)$
AH2	$CA/2(8)$	2/AH2	$CA/6(2)$
AJ2	$CR/5(3)$	2/AJ2	$CR/6(12)$
AK2	$C9/2(9)$	2/AK2	$C9/7(5),15(4)$
AL2	$CB/3(8)$	2/AL2	$CB/7(13)$

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
AM2	C2/2(9)	2/AM2	C2/8(1)
AN2	C5/1(5)	2/AN2	C5/10(1),161
AP2	CC/1(8)	2/AP2	CC/8(2)
AR2	CT/5(11)	3/AR2	CT/42(5)
AS2	CH/1(3)	2/AS2	CH/19(3)
AT2	CP/1(6)	3/AT2	CP/34(13)
AU2	CN/2(11)	3/AU2	CN/34(2)
AV2	CD/1(11)	2/AV2	CD/14(1)
BA1	B1/E(2)	2/BA1 3/BA1	B1/10(9) B1/34(1)
BC1	B2/E(4)	2/BC1	B2/10(12)
BD1	B3/E(6)	2/BD1	B3/10(3)
BE1	B5/E(8)	2/BE1	B5/11(4)
BF1	B9/E(10)	2/BF1	B9/11(5)
BH1	B14/E(12)	2/BH1	B14/15(10)
BJ1	A1/G(2)	2/BJ1	A1/24(1)
BK1	A3/G(4)	2/BK1	A3/24(2)
BL1	A5/G(8)	2/BL1	A5/17(13)
BM1	A7/G(10)	3/BM1	A7/36(12)
BN1	A9/G(12)	2/BN1	A9/30(10)
BP1	A11/F(4)	2/BP1	A11/27(13)
BR1	A14/F(6)	2/BR1	A14/23(2)
BS1	A16/F(8)	2/BS1	A16/14(4)
BU1	A17/F(10)	2/BU1	A17/15(13)
BV1	C4/D(8)	2/BV1	C4/10(2)
BB2	CI/2(3)	2/BB2	CI/19(10)
BD2	CQ/2(6)	3/BD2	CQ/38(10)

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
BE2	CJ/3(6)	2/BE2	CJ/16(1)
BF2	CK/3(11)	2/BF2	CK/23(10)
BH2	CL/4(6)	2/BH2	CL/26(2)
BJ2	CV/4(11)	3/BE2	CV/43(9)
BK2	CU/5(8)	3/BE1	CU/42(13)
BL2	--/M(4)	4/BU2	--/67(3)
BM2	G2/A(19)	3/AP2	G2/44(3)
BN2	IR <sub>3</sub> /K(9)	7/CN2	IR <sub>3</sub> /144(12)
BP2	CE/1(1)	2/BP2	CE/15(2)
BR2	$\overline{IR}_1$ /K(14)	7/CM2	$\overline{IR}_1$ /147(12)
BS2	$\overline{IR}_0$ /K(1)	7/CL2	$\overline{IR}_0$ /147(13)
BT2	IR <sub>0</sub> /K(16)	4/BH2	IR <sub>0</sub> /64(9)
BU2	DIR <sub>3</sub> /A(23)	4/BT2	DIR <sub>3</sub> /66(3)

## Module 2. Instruction Register Decoder

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
AP1	--/30(4)	3/AP1	--/33(3)
AR1	C3/7(2)	1/AR1	C3/1(4)
AS1	BH/12(3)	3/AS1	BH/34(5)
AU1	V/15(3)	3/AU1	V/34(9)
AV1	X/32(8)	3/AV1	X/40(4)
AB2	C6/6(13)	3/AB2 1/AB2	C6/43(4) C6/D(12)
AD2	C11/6(1),9(13)	1/AD2	C11/2(2)
AE2	C10/6(5)	1/AE2	C10/3(9)
AF2	C8/6(4)	1/AF2	C8/5(5)
AH2	CA/6(2)	1/AH2 3/AH2	CA/2(8) CA/33(5)
AJ2	CR/6(12)	1/AJ2 3/AJ2	CR/5(3) CR/38(13)
AK2	C9/7(5),15(4)	1/AK2 3/AK2	C9/2(1) C9/38(4)
AL2	CB/7(13)	1/AL2 3/AL2	CB/3(8) CB/33(13)
AM2	C2/8(1)	1/AM2	C2/2(9)
AN2	C5/10(1),16(13)	1/AN2	C5/1(5)
AP2	CC/8(2)	1/AP2	CC/1(8)
AR2	CT/6(11)	3/AN2	CT/47(13)
AS2	CH/19(3)	1/AS2	CH/1(3)
AV2	CD/14(1)	1/AV2	CD/1(11)
BA1	B1/10(9)	1/BA1	B1/E(2)
BC1	B2/10(12)	1/BC1 3/BC1	B2/E(4) B2/33(12)
BD1	B3/10(13)	1/BD1 3/BD1	B3/E(6) B3/33(4)

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
BE1	B5/11(4)	1/BE1	B5/E(8)
BF1	B9/11(5)	1/BF1	B9/E(10)
BH1	B14/15(10)	1/BH1 3/BH1	B14/E(12) B14/39(4)
BJ1	A1/24(1)	1/BJ1 3/BJ1	A1/G(2) A1/46(5)
BK1	A3/24(2)	1/BK1 3/BK1	A3/G(4) A3/36(1)
BL1	A5/17(13)	1/BL1 3/BL1	A5/G(8) A5/36(10)
BM1	BA/13(3)	3/BV1	BA/47(1)
BN1	A9/30(10)	1/BN1	A9/G(12)
BP1	A11/27(13)	1/BP1 3/BP1	A11/F(4) A11/41(4)
BR1	A14/23(2)	1/BR1 3/BR1	A14/F(6) A14/33(2)
BS1	A16/14(4)	1/BS1 3/BS1	A16/F(8) A16/33(10)
BU1	A17/15(13)	1/BU1 3/BU1	A17/F(10) A17/38(2)
BV1	C4/10(2)	1/BV1	C4/D(8)
BB2	CI/19(10)	1/BB2	CI/2(3)
BD2	Y/22(5)	3/BF2	Y/49(2)
BE2	CJ/16(1)	1/BE2	CJ/3(6)
BF2	CK/23(10)	1/BF2	CK/3(11)
BH2	CL/26(2)	1/BH2	CL/4(6)
BJ2	BJ/11(11)	3/BJ2	BJ/38(5)
BK2	CS/10(6)	3/BK2	CS/39(5)
BL2	BK/11(6)	3/BN2	BK/39(13)
BM2	B5/11(4)	3/BM2	B5/39(9)

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
BN2	DOS/27(1)	3/BU2	DOS/44(2)
BP2	CE/15(2)	3/BP2 1/BP2	CE/34(4) CE/1(1)
BR2	F11/13(11)	3/AD2	F11/50(4)
BS2	F10/22(11)	3/AE2	F10/50(10)
BT2	F9/28(11)	3/AF2	F9/50(3)
BU2	F8/31(11)	3/AM2	F8/50(11)
BV2	--/30(2)	3/BV2	--/33(8)

### Module 3. Instruction Register Decoder and MPC

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
AA1	$\overline{B}_{11}/53(3)$		
AB1	$\overline{B}_{10}/53(6)$		
AC1	$\overline{B}_9/53(11)$		
AD1	$\overline{B}_8/53(8)$		
AE1	$\overline{B}_7/54(3)$		
AF1	$\overline{B}_6/54(6)$		
AH1	$\overline{B}_5/54(11)$		
AJ1	$\overline{B}_4/54(8)$		
AK1	$\overline{B}_3/55(3)$		
AL1	$\overline{B}_2/55(6)$		
AM1	$\overline{B}_1/55(11)$		
AN1	$\overline{B}_0/55(8)$		
AP1	--/33(3)	2/AP1	--/30(4)
AR1	SGD/50(8)	7/CU1	SGD/150(6)
AS1	BH/34(5)	2/AS1	BH/12(3)
AU1	V/34(9)	2/AU1	V/15(3)
AV1	X/40(4)	2/AV1	X/32(8)
AB2	C6/43(4)	2/AB2	C6/6(13)
AD2	F11/50(4)	2/BR2	F11/13(11)
AE2	F10/50(10)	2/BS2	F10/22(11)
AF2	F9/50(3)	2/BT2	F9/28(11)
AH2	CA/33(5)	2/AH2	CA/6(2)
AJ2	CR/38(13)	2/AJ2	CR/6(12)
AK2	C9/38(4)	2/AK2	C9/7(5),15(.
AL2	CB/33(13)	2/AL2	CB/7(13)

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
AM2	F8/50(11)	2/BU2	F8/31(11)
AN2	CW/47(13)	2/AR2	CW/6(11)
AP2	G2/44(3)	1/BM2	G2/C(19)
AR2	CT/42(5)	1/AR2	CT/5(11)
AS2	SGA/55(9)	16/AK1	SGA/158(8)
AT2	CP/34(13)	1/AT2	CP/1(6)
AU2	CN/34(2)	1/AU2	CN/2(11)
AV2	K/45(11)	4/BV1	K/66(8)
BA1	B1/34(1)	1/BA1	B1/E(2)
BB1	MC/50(13)		
BC1	B2/33(12)	2/BC1	B2/10(12)
BD1	B3/33(4)	2/BD1	B3/10(13)
BE1	CU/42(13)	1/BK2	CU/5(8)
BF1	F6/37(11)		
BH1	B14/39(4)	2/BH1	B14/15(10)
BJ1	A1/46(5)	2/BJ1 1/BJ1	A1/24(1) A1/G(2)
BK1	A3/36(1)	2/BK1	A3/36(1)
BL1	A5/36(10)	2/BL1	A5/36(10)
BM1	A7/36(12)	1/BM1	A7/G(10)
BN1	A9/41(2)	1/BN1	A9/G(12)
BP1	A11/41(4)	2/BP1	A11/27(13)
BR1	A14/33(2)	2/BR1	A14/23(2)
BS1	A16/33(10)	2/BS1	A16/14(4)
BU1	A17/38(2)	2/BU1	A17/15(13)
BV1	BA/47(1)	2/BM1	BA/13(3)
BD2	CQ/38(10)	1/BD2	CQ/2(6)

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
BE2	CV/43(9)	1/BJ2	CV/4(11)
BF2	Y/49(2)	2/BD2	Y/22(5)
BJ2	BJ/38(5)	2/BJ2	BJ/11(11)
BK2	CS/39(5)	2/BK2	CS/10(6)
BL2	F7/37(6)		
BM2	B5/39(9)	2/BM2	B5/11(4)
BN2	BK/39(13)	2/BL2	BK/11(6)
BP2	CE/34(4)	2/BP2	CE/15(2)
BR2	F5/44(8)		
BS2	F/44(1)	7/CU2	F/147(3)
BT2	F4/45(6)		
BU2	DOS/44(2)	2/BN2 4/AF2	DOS/27(1) DOS/70(6)
BV2	--/33(8)	2/BV2	--/30(2)

# Module 4. Miscellaneous Circuits

<u>PIn</u>	<u>Signal/IC</u>	<u>Module/PIn</u>	<u>Signal/IC</u>
AA1	$\overline{B}_{11}/58(2)$		
AB1	$\overline{B}_{10}/58(4)$		
AC1	$\overline{B}_9/58(6)$		
AD1	$\overline{B}_8/58(8)$		
AE1	$\overline{B}_7/58(10)$		
AF1	$\overline{B}_6/58(12)$		
AH1	$\overline{B}_5/57(2)$		
AJ1	$\overline{B}_4/57(4)$		
AK1	$\overline{B}_3/57(6)$		
AL1	$\overline{B}_2/57(8)$		
AM1	$\overline{B}_1/57(10)$		
AN1	$\overline{B}_0/57(12)$		
AP1	SGE/62(1)	7/DK2	SGE/148(8)
AR1	CD30/62(4)	5/BU2	CD30/89(3)
AS1	SGF/62(5)	7/DF2	SGF/148(6)
AU1	--/63(3)	7/DF1	--/154(3)
AV1	--/65(6)	8/AU1	--/167(5)
AB2	CD63/68(1)	6/BK2	CD63/100(11)
AD2	LOAD/56(2)	7/DN2	LOAD/136(10)
AE2	--/70(5)	7/CP2	--/144(11)
AF2	DOS/70(6)	3/BU2	DOS/44(2)
AH2	CD31/56(4)	5/BV2	CD31/89(6)
AJ2	DEP/56(1)	7/CN1	DEP/160(8)
AK2	BUSM/62(6)	11/BN1	BUSM
AL2	DONE/62(8)	11/BD1	DONE

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
AM2	--/62(11)	8/AF2	--/166(11)
AN2	--/63(5)	7/DE2	--/149(3)
AP2	CLK/56(12)	7/DK1	CLK/158(1)
AR2	CD64/68(5)	6/BL2	CD64/101(3)
AS2	ABUS/68(3)	10/BN1	ABUS
AT2	BBUS/68(6)	10/BJ1	BBUS
AU2	MAD/63(8)	11/BF2	MAD
BD1	SR <sub>7</sub> /60(2)		
BE1	SR <sub>6</sub> /60(5)		
BF1	SR <sub>5</sub> /60(8)		
BH1	SR <sub>4</sub> /60(11)		
BJ1	SR <sub>3</sub> /61(2)		
BK1	SR <sub>2</sub> /61(5)		
BL1	SR <sub>1</sub> /61(8)		
BM1	SR <sub>0</sub> /61(11)		
BN1	SR <sub>11</sub> /59(2)		
BP1	SR <sub>10</sub> /59(5)		
BR1	SR <sub>9</sub> /59(8)		
BS1	SR <sub>8</sub> /59(11)		
BU1	CD73/66(10)	6/BU2	CD73/102(11)
BV1	K/66(8)	3/AV2	K/45(11)
BB2	CD5/65(9)	5/BH1	CD5/83(6)
BD2	CD22/65(4)	5/BM2	CD22/87(8)
BE2	--/65(11)	8/AH2	--/167(12)
BF2	MBUS/63(12)	11/BK1	MBUS
BH2	IR <sub>0</sub> /64(9)	1/BT2	IR <sub>0</sub> /K(16)

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
BJ2	IR <sub>1</sub> /63(11)	1/AS1	IR <sub>1</sub> /K(15)
BK2	IR <sub>2</sub> /64(12)	1/AP1	IR <sub>2</sub> /K(10)
BL2	CD40/64(13)	6/BA1	CD40/96(3)
BM2	CD56/65(2)	6BD2	CD56/99(8)
BN2	--/65(3)	7/BV1	--/136(9)
BP2	cd15/65(13)	5/BF2	CD15/86(6)
BR2	--/66(1)	7/DT2	--/151(4)
BS2	IR <sub>3</sub> /66(2)	7/CN2	IR <sub>3</sub> /144(12)
BT2	DIR <sub>3</sub> /66(3)	1/BU2	DIR <sub>3</sub> /A(23)
BU2	--/67(3)	1/BL2	--/M(4)
BV2	--/67(2)	7/DJ2	--/149(5)

# Module 5. MIR and Field A Decoder

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
AA1	$\overline{B}_{11}/84(13)$		
AB1	$\overline{B}_{10}/84(11)$		
AC1	$\overline{B}_9/84(9)$		
AD1	$\overline{B}_8/84(5)$		
AE1	$\overline{B}_7/84(3)$		
AF1	$\overline{B}_6/84(1)$		
AH1	$\overline{B}_5/75(13)$		
AJ1	$\overline{B}_4/75(11)$		
AK1	$\overline{B}_3/75(9)$		
AL1	$\overline{B}_2/75(5)$		
AM1	$\overline{B}_1/75(3)$		
AN1	$\overline{B}_0/75(1)$		
AP1	CD32/89(8)		
AR1	CD33/89(11)	6/AF2	CD33/107(12)
AS1	CD34/91(3)	7/AU1	CD34/136(1)
AU1	$\overline{CD35}/92(8)$	7/CJ2 9/BK2	$\overline{CD35}/147(9)$ $\overline{CD35}$
AV1	$\overline{CD36}/92(10)$	7/CK2 9/AL2	$\overline{CD36}/147(10)$ $\overline{CD36}$
AD2	$\overline{CD13}/92(6)$	11/BM1	$\overline{CD13}$
AP2	CD37/91(11)	8/AP1	CD37/200(9)
AS2	SGG/79(14)	7/CV2	SGG/53(6)
AU2	--/78(16)	6/AM1	--/110(13)
BA1	CD0/82(3)		
BC1	CD1/82(6)	7/AF2	CD1/155(2)
BD1	CD2/82(8)	6/AN2	CD2/104(2)

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
BE1	CD3/92(2)	9/AR2	CD3
BF1	CD4/83(3)	7/CR2	CD4/147(2)
BH1	CD5/83(6)	4/BB2	CD5/65(9)
BJ1	CD6/83(8)	6/AM2	CD6/104(5)
BK1	CD7/92(4)	9/BB2	CD7
BL1	CD10/85(3)	6/AL2	CD10/104(10)
BM1	CD11/92(12)	9/BL2	CD11
BN1	SGD/76(4)	7/CU1	SGD/150(6)
BP1	--/77(15)	6/BP1	--/109(20)
BR1	--/77(16)	6/BR1	--/109(21)
BS1	--/78(9)	6/BS1	--/109(22)
BU1	--/78(10)	6/BU1	--/109(23)
BV1	--/78(15)	6/BV1	--/110(3)
BB2	CD12/85(8)	6/AK2	CD12/104(13)
BD2	CD13/85(11)	7/DE1	CD13/154(2)
BE2	CD14/86(3)	6/AJ2	CD14/106(2)
BF2	CD15/86(6)	4/BP2	CD15/65(13)
BH2	CD16/86(8)	6/AH2	CD16/106(5)
BJ2	CD17/86(11)	7/BU2	CD17/139(5)
BK2	CD20/87(3)	6/AE2	CD20/106(10)
BL2	CD21/87(6)	7/AV1	CD21/136(2)
BM2	CD22/87(8)	4/BD2	CD22/65(4)
BN2	CD23/87(11)	6/AD2	CD23/106(13)
BP2	CD24/88(3)	8/AJ2	CD24/167(13)
BR2	CD25/88(6)	6/AB2	CD25/108(2)
BS2	CD26/88(8)	7/AB2	CD26/141(13)

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
BT2	CD27/88(11)	8/AB2	CD27/167(4)
BU2	CD30/89(3)	4/AR1	CD30/65(10)
BV2	CD31/89(6)	4/AH2	CD31/56(4)

# Module 6. Field B Decoder and Increment Program Counter

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
AK1	CD66/105(12)	11/BE2	CD66
AL1	SGG/110(14)	5/AS2	SGG/79(14)
AM1	110(13)	5/AU2	78(16)
AN1	KRIRF/111(2)		
AP1	PRIRF/111(1)		
AR1	IPC/111(11)	7/BP1	IPC/144(4)
AS1	KRF/106(9)		
AW1	AC <sub>0</sub> /106(1)	7/BS2	AC <sub>0</sub> /117(12)
AV1	AC/104(9)	7/BR1	AC/138(8)
AB2	CD25/108(2)	5/BR2	CD25/88(6)
AD2	CD23/106(13)	5/BN2	CD23/87(11)
AE2	CD20/106(10)	5/BK2	CD20/87(3)
AF2	CD33/107(12)	5/AR1	CD33/89(11)
AH2	CD16/106(5)	5/BH2	CD16/86(8)
AJ2	CD14/106(2)	5/BE2	CD14/86(3)
AK2	CD12/104(13)	5/BB2	CD12/85(8)
AL2	CD10/104(10)	5/BL1	CD10/85(3)
AM2	CD6/104(5)	5/BJ1	CD6/83(8)
AN2	CD2/104(2)	5/BD1	CD2/82(8)
AP2	L/104(1)	7/CE2	L/143(15)
AR2	PPF/106(12)		
AS2	CD55/99(6)		
AT2	CD77/103(11)		
AU2	CD76/112(12)	9/BT2	CD76
AV2	CD75/103(6)	7/DH1	CD75/151(9)

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
BA1	CD40/96(3)	4/BL2	CD40/64(13)
BB1	CD41/96(6)	7/BU1	CD41/140(1)
BC1	CD42/96(8)	7/AE2	CD42/141(2)
BD1	CD43/96(11)	7/AD2	CD43/142(9)
BE1	CD44/97(3)	7/CH2	CD44/141(10)
BF1	$\overline{\text{CD45}}/112(2)$	9/BJ1	$\overline{\text{CD45}}$
BH1	CD46/97(8)	7/CD2	CD46/141(5)
BJ1	CD47/97(11)		
BK1	CD50/98(3)	7/AP1	CD50/135(9)
BL1	CD51/98(6)	7/DM1	CD51/157(4)
BM1	$\overline{\text{CD52}}/112(4)$	9/BU2	$\overline{\text{CD52}}$
BN1	$\overline{\text{CD53}}/112(6)$	9/AV2	$\overline{\text{CD53}}$
BP1	--/109(20)	5/BP1	--/77(15)
BR1	--/109(21)	5/BR1	--/77(16)
BS1	--/109(22)	5/BS1	--/78(9)
BU1	--/109(23)	5/BU1	--/78(10)
BV1	--/110(3)	5/BV1	--/78(15)
BB2	CD54/99(3)		
BD2	CD56/99(8)	4/BM2	CD56/65(2)
BE2	CD57/99(11)		
BF2	CD60/100(3)		
BH2	CD61/100(6)		
BJ2	CD62/100(8)		
BK2	CD63/100(11)	4/AB2	CD63/68(1)
BL2	CD64/101(3)	4/AR2	CD64/68(5)
BM2	CD65/101(6)	8/AS1	CD65/167(2)
BN2	CD66/101(8)		

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
BP2	CD67/101(11)	7/DD2	CD67/149(2)
BR2	CD70/102(3)	7/DH2	CD70/149(5)
BS2	CD71/102(6)	7/AR1	CD71/135(11)
BT2	CD72/102(8)	7/AS1	CD72/139(4)
BU2	CD73/102(11)	4/BU1	CD73/66(10)
BV2	CD74/103(3)	8/AR1	CD74/167(1)

# Module 7. Accumulator, Program Counter, Control State Generator

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
AA1	$\overline{B}_{11}/114(13)$		
AB1	$\overline{B}_{10}/114(11)$		
AC1	$\overline{B}_9/114(9)$		
AD1	$\overline{B}_8/114(5)$		
AE1	$\overline{B}_7/114(3)$		
AF1	$\overline{B}_6/114(1)$		
AH1	$\overline{B}_5/113(13)$		
AJ1	$\overline{B}_4/113(11)$		
AK1	$\overline{B}_3/113(9)$		
AL1	$\overline{B}_2/113(5)$		
AM1	$\overline{B}_1/113(3)$		
AN1	$\overline{B}_0/113(1)$		
AP1	CD50/135(9)	6/BK1	CD50/98(3)
AR1	CD71/135(11)	6/BS2	CD71/102(6)
AS1	CD72/139(4)	6/BT2	CD72/102(8)
AU1	CD34/136(1)	5/AS1	CD34/91(3)
AV1	CD21/136(2)	5/BL2	CD21/87(6)
AB2	CD26/141(13)	5/BS2	CD26/88(8)
AD2	CD43/142(9)	6/BD1	CD43/96(11)
AE2	CD42/141(2)	6/BC1	CD42/96(8)
AF2	CD1/155(2)	5/BC1	CD1/82(6)
BA1'	$\overline{B}_{11}/134(12)$		
BB1	$\overline{B}_{10}/134(10)$		
BC1	$\overline{B}_9/134(8)$		
BD1	$\overline{B}_8/134(6)$		

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
BE1	$\overline{B}_7/134(4)$		
BF1	$\overline{B}_6/134(2)$		
BH1	$\overline{B}_5/133(12)$		
BJ1	$\overline{B}_4/133(10)$		
BK1	$\overline{B}_3/133(8)$		
BL1	$\overline{B}_2/133(6)$		
BM1	$\overline{B}_1/133(4)$		
BN1	$\overline{B}_0/133(2)$		
BP1	IPC/144(4)	6/AR1	IPC/111(11)
BR1	AC/138(8)	6/AV1	AC/104(9)
BS1	IR <sub>4</sub> /140(2)	1/AV1	IR <sub>4</sub> /1(16)
BU1	CD41/140(1)	6/BB1	CD41/96(6)
BV1	--/136(9)	4/BN2	--/65(3)
BS2	AC <sub>0</sub> /117(12)	6/AU1	AC <sub>0</sub> /106(1)
BU2	CD17/139(5)	5/BJ2	CD17/86(11)
CN1	DPOS/159(6)	4/AJ2	DPOS/56(1)
CB2	OF/146(3)	9/AP1	OF
CD2	CD46/141(5)	6/BH1	CD46/97(8)
CE2	L/143(15)	6/AP2	L/104(1)
CF2	$\overline{MC}/142(3)$	9/BB1	$\overline{MC}$
CH2	CD44/141(10)	6/BE1	CD44/97(3)
CJ2	$\overline{CD35}/147(9)$	5/AU1	$\overline{CD35}/92(8)$
CK2	$\overline{CD36}/147(10)$	5/AV1	$\overline{CD36}/92(10)$
CL2	$\overline{IR}_0/147(13)$	1/BS2	$\overline{IR}_0/K(1)$
CM2	$\overline{IR}_1/147(12)$	1/BR2	$\overline{IR}_1/K(14)$
CN2	IR <sub>3</sub> /144(12)	1/BN2	IR <sub>3</sub> /K(9)
		4/BS2	IR <sub>3</sub> /66(2)
CP2	144(11)	4/AE2	70(5)
CR2	CD4/147(2)	5/BF1	CD4/83(3)
CU2	F/147(3)	3/BS2	F/44(1)
DE1	CD13/154(2)	5/BD2	CD13/85(11)
DF1	--/154(3)	4/AU1	--/63(3)

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
DH1	CD75/151(9)	6/AV2	CD75/103(6)
DJ1	$\overline{\text{CD75}}$ /151(8)	11/BF1	$\overline{\text{CD75}}$
DL1	--/149(11)		
DN1	DEP/161(10)	5/AF2	DEP/93(6)
DS1	SGD/154(1)	16/AN1	SGD/172(13)
DU1	$\overline{\text{MC}}$ /161(13)	12/AJ1	$\overline{\text{MC}}$ /219(6)
DD2	CD67/149(2)	6/BP2	CD67/101(11)
DE2	--/149(3)	4/AN2	--/63(5)
DH2	CD70/149(5)	6/BR2	CD70/102(3)
DJ2	--/149(5)	4/BV2	--/67(2)
DL2	$\overline{\text{DEP}}$ /161(5)	11/BH2	$\overline{\text{DEP}}$
DM2	$\overline{\text{DPOS}}$ /159(1)	11/BH1	$\overline{\text{DPOS}}$
DN2	LOAD/136(10)	12/AK1	LOAD/219(1)
DT2	--/151(4)	4/BR2	--/66(1)
CA1	PC <sub>0</sub> /124(12)	12/AD2	PC <sub>0</sub> /224(1)
CB1	PC <sub>1</sub> /124(2)	12/AE2	PC <sub>1</sub> /224(3)
CC1	PC <sub>2</sub> /124(9)	12/AF2	PC <sub>2</sub> /224(5)
CD1	PC <sub>3</sub> /124(5)	12/AH2	PC <sub>3</sub> /224(13)
CE1	PC <sub>4</sub> /125(12)	12/AJ2	PC <sub>4</sub> /224(11)
CF1	PC <sub>5</sub> /125(2)	12/AK2	PC <sub>5</sub> /224(9)
CH1	PC <sub>6</sub> /125(9)	12/AL2	PC <sub>6</sub> /225(2)
CJ1	PC <sub>7</sub> /125(5)	12/AM2	PC <sub>7</sub> /225(3)
CK1	PC <sub>8</sub> /126(12)	12/AN2	PC <sub>8</sub> /225(5)
CL1	PC <sub>9</sub> /126(3)	12/AP2	PC <sub>9</sub> /225(13)
CS2	PC <sub>10</sub> /126(9)	12/AR2	PC <sub>10</sub> /225(11)
CM1	PC <sub>11</sub> /126(5)	12/AS2	PC <sub>11</sub> /225(9)

Module 8. MAR, MMAR, MBR, MMBR

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
AA1	$\bar{B}_{11}/176(13)$		
AB1	$\bar{B}_{10}/176(11)$		
AC1	$\bar{B}_9/176(9)$		
AD1	$\bar{B}_8/176(5)$		
AE1	$\bar{B}_7/176(3)$		
AF1	$\bar{B}_6/176(1)$		
AH1	$\bar{B}_5/177(13)$		
AJ1	$\bar{B}_4/177(11)$		
AK1	$\bar{B}_3/177(9)$		
AL1	$\bar{B}_2/177(5)$		
AM1	$\bar{B}_1/177(3)$		
AN1	$\bar{B}_0/177(1)$		
AP1	CD37/200(9)	5/AP2	CD37/91(11)
AR1	CD74/167(1)	5/BV2	CD74/103(3)
AS1	CD65/167(2)	5/BM2	CD65/101(6)
AU1	--/167(5)	4/AV1	--/65(6)
AB2	CD27/167(4)	5/BT2	CD27/88(11)
AD2	SGAI/179(14)	16/AK2	SGAI
AF2	--/166(11)	4/AM2	--/62(11)
AH2	--/167(12)	4/BE2	--/65(11)
AJ2	CD24/167(13)	5/BD2	CD24/88(3)
AK2	SGCI/182(14)	16/AM2	SGCI

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
BE1	MBRSB/172(2)		
BF1	SGD/172(3)	3/AR1	SGD/50(8)
BH1	SGDB/172(4)		
BJ1	IRG/172(5)	4/BB1	IRG/67(3)
BK1	IRGB/172(6)		
BL1	SGC/172(13)	7/DS1	SGC/161(10)
BM1	SGCB/172(12)		
BN1	SGA/172(11)	7/CS1	SGA/149(8)
BP1	SGAB/172(10)		
BR1	SGA/172(10)	3/AS2	SGA/55(9)
BS1	SGA(Buf)/172(9)	13/BS2	SGA(Buf)/236(13)
CA1	$\bar{B}_{11}$ /184(12)		

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
CB1	$\overline{B}_{10}/184(10)$		
CC1	$\overline{B}_9/184(8)$		
CD1	$\overline{B}_8/184(6)$		
CE1	$\overline{B}_7/184(4)$		
CF1	$\overline{B}_6/184(2)$		
CH1	$\overline{B}_5/185(12)$		
CJ1	$\overline{B}_4/185(10)$		
CK1	$\overline{B}_3/185(8)$		
CL1	$\overline{B}_2/185(6)$		
CM1	$\overline{B}_1/185(4)$		
CN1	$\overline{B}_0/185(2)$		
CP1	SGF/200(2)	7/DF2	SGF/148(6)
DN2	$\overline{TR}_4/200(10)$	1/AU1	$\overline{TR}_4/L(1)$

# Module 9. General Purpose Arithmetic Unit (M7300)

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
AA1	$\overline{B}_{11}$		
AB1	$\overline{B}_{10}$		
AC1	$\overline{B}_9$		
AD1	$\overline{B}_8$		
AE1	$\overline{B}_7$		
AF1	$\overline{B}_6$		
AH1	$\overline{B}_5$		
AJ1	$\overline{B}_4$		
AK1	$\overline{B}_3$		
AL1	$\overline{B}_2$		
AM1	$\overline{B}_1$		
AN1	$\overline{B}_0$		
AP1	OF (AN1)	7/CB2	OF/146(3)
AR1	Bus Term (AP1)		
AS1	Bus Term (AR1)		
AU1	Bus Term (AS1)		
AL2	$\overline{CD36}$	5/AV1	$\overline{CD36}/92(10)$
AM2	Vcc		
AN2	Vcc		
AP2	Vcc		
AR2	$\overline{CD3}$	5/BE1	$\overline{CD3}/92(2)$
AS2	Vcc		
AT2	Vcc		
AU2	Vcc		
AV2	$\overline{CD53}$	6/BN1	$\overline{CD53}/112(6)$

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
BB1	$\overline{MC}$	7/CF2	MC/142(3)
BJ1	$\overline{CD45}$	6/BF1	$\overline{CD45}/112(2)$
BK1	Vcc		
BL1	Vcc		
BM1	Vcc		
BN1	Vcc		
BB2	$\overline{CD7}$	5/BK1	$\overline{CD7}/92(4)$
BD2	Vcc		
BE2	Vcc		
BF2	Vcc		
BK2	$\overline{CD35}$	5/AU1	$\overline{CD35}/92(8)$
BP2	Vcc		
BR2	Vcc		
BS2	Vcc		
BT2	$\overline{CD76}$	6/AU2	$\overline{CD76}/112(12)$
BU2	$\overline{CD52}$	6/BM1	$\overline{CD52}/112(4)$

# Module 10. General Purpose Arithmetic Unit (M7301)

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
AA1	$\overline{B}_{11}$		
AB1	$\overline{B}_{10}$		
AC1	$\overline{B}_9$		
AD1	$\overline{B}_8$		
AE1	$\overline{B}_7$		
AF1	$\overline{B}_6$		
AH1	$\overline{B}_5$		
AJ1	$\overline{B}_4$		
AK1	$\overline{B}_3$		
AL1	$\overline{B}_2$		
AM1	$\overline{B}_1$		
AN1	$\overline{B}_0$		
AP1	Bus Term (AN1)		
AR1	Bus Term (AP1)		
AS1	Bus Term (AR1)		
AU1	Bus Term (AS1)		
BA1	Vcc		
BB1	$\overline{MC}$	9/BB1	$\overline{MC}$
BC1	Vcc		
BD1	Vcc		
BE1	Vcc		
BJ1	BBUS	4/AT2	BBUS/68(3)
BK1	Vcc		
BL1	Vcc		
BM1	Vcc		
BN1	ABUS	4/AS2	ABUS/68(6)

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
BP1	Vcc		
BR1	Vcc		
BS1	Vcc		
BJ2	Vcc		
BK2	Vcc		
BL2	Vcc		
BM2	Vcc		
BN2	Vcc		
BP2	Vcc		
BR2	Vcc		
BS2	Vcc		

Module 11. Memory (M7319)

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
AA1	$\overline{B}_{11}$		
AB1	$\overline{B}_{10}$		
AC1	$\overline{B}_9$		
AD1	$\overline{B}_8$		
AE1	$\overline{B}_7$		
AF1	$\overline{B}_6$		
AH1	$\overline{B}_5$		
AJ1	$\overline{B}_4$		
AK1	$\overline{B}_3$		
AL1	$\overline{B}_2$		
AM1	$\overline{B}_1$		
AN1	$\overline{B}_0$		
AP1	Bus Term	10/AP1	
AR1	Bus Term	10/AR1	
AS1	Bus Term	10/AS1	
AU1	Bus Term	10/AU1	
BB1	$\overline{MC}$	9/BB1	$\overline{MC}$
BD1	DONE	4/AL2	DONE/62(8)
BE1	Grnd		
BF1	$\overline{CD75}$	7/DJ1	$\overline{CD75}/151/(8)$
BH1	$\overline{DPOS}$	7/DM2	$\overline{DPOS}/159(1)$

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
BK1	MBUS	4/BF2	MBUS/63(12)
BL1	SGCI	7/dd1	SGCI/151(10)
BM1	$\overline{\text{CD13}}$	5/AD2	$\overline{\text{CD13}}/92(6)$
BN1	BUSM	4/AK2	BUSM/62(6)
BU1	-15 VDC		
BD2	SGBI	7/DA1	SGBI/151(12)
BE2	$\overline{\text{CD66}}$	6/AK1	$\overline{\text{CD66}}/105(12)$
BF2	MAD	4/AU2	MAD/63(8)
BH2	$\overline{\text{DEP}}$	7/DL2	$\overline{\text{DEP}}/160(11)$

# Module 12. Control Panel Interface Board 1

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
AA1	MAR0/201(4)	8/AM2	MAR0/178(5)
AB1	MAR1/202(4)	8/AN2	MAR1/178(7)
AC1	MAR2/212(4)	8/AP2	MAR2/178(9)
AD1	MAR3/211(4)	8/AR2	MAR3/178(11)
AE1	MAR4/213(4)	8/AS2	MAR4/178(17)
AF1	MAR5/214(4)	8/AT2	MAR5/178(19)
AH1	CLKP2		
AJ1	CLKT2		
AK1	LOAD/219(11)	7/DN2	LOAD/136(10)
AL1	DEP/219(8)	7/DN1	DEP/161(10)
AM1	CLKS2		
AN1	CLKR2		
AS1	CONTO/201(11)	13/AS1	CONTO/261(12)
AT1	GND		
AU1	CONT1/201(10)	13/AU1	CONT1/261(10)
AV1	CONT2/201(9)	13/AV1	CONT2/261(8)
AA2	Vcc		
AB2	CS4/228(1)		
AC2	GND		
AD2	PC0/224(1)	7/CA1	PC0/124(12)
AE2	PC1/224(3)	7/CB1	PC1/124(2)
AF2	PC2/224(5)	7/CC1	PC2/124(9)

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
AH2	PC3/224(13)	7/CD1	PC3/124(5)
AJ2	PC4/224(11)	7/CE1	PC4/125(12)
AK2	PC5/224(9)	7/CF1	PC5/125(2)
AL2	PC6/225(1)	7/CH1	PC6/125(9)
AM2	PC7/225(3)	7/CJ1	PC7/125(5)
AN2	PC8/225(5)	7/CK1	PC8/126(12)
AP2	PC9/225(13)	7/CL1	PC9/126(2)
AR2	PC10/225(11)	7/CS2	PC10/126(9)
AS2	PC11/225(9)	7/CM1	PC11/126(5)
AT2	L/228(9)	7/CE2	L/143(15)
AU2	LD/228(8)		
AV2	$\overline{MC}$ /231(13)	7/CF2	$\overline{MC}$ /142(3)
BA1	SR0/221(15)	4/BM1	SR0/61(11)
BB1	SR1/221(4)	4/BL1	SR1/61(8)
BC1	SR2/217(15)	4/BK1	SR2/61(5)
BD1	SR3/217(11)	4/BJ1	SR3/61(2)
BE1	SR4/220(15)	4/BH1	SR4/60(11)
BF1	SR5/220(11)	4/BF1	SR5/60(8)
BH1	F4/231(11)	3/BT2	F4/45(6)
BJ1	F5/231(3)	3/BR2	F5/44(8)
BK1	F6/231(10)	3/BF1	F6/37(11)
BL1	F7/231(4)	3/BL2	F7/37(6)
BM1	MPC6/231(9)		

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
BN1	MPC7/231(5)		
BP1	CLKN2		
BR1	NANO2/208(10)	13/BR1	NANO2/238(8)
BS1	BUS4/229(10)		
BT1	GND		
BU1	BUS5/229(8)		
BV1	IR4/213(12)	13/BV1	IR4/253(10)
BA2	Vcc		
BB2	IR5/214(12)	13/BB2	IR5/253(9)
BC2	GND		
BD2	$\overline{\text{BUS0}}$ /229(1)	8/CN1	
BE2	$\overline{\text{BUS1}}$ /229(3)	8/CM1	
BF2	$\overline{\text{BUS2}}$ /229(5)	8/CL1	
BH2	$\overline{\text{BUS3}}$ /229(13)	8/CK1	
BJ2	$\overline{\text{BUS4}}$ /229(11)	8/CJ1	
BK2	$\overline{\text{BUS5}}$ /229(9)	8/CH1	
BL2	OSG2/228(10)		
BM2	IRGB/215(13)	8/BK1	IRGB/172(6)
BN2	MPG/231(1)	3/BH2	MPG/48(8)
BP2	MBRSB/203(13)	8/BE1	MBRSB/172(2)
BR2	SGDB/209(13)	8/BH1	SGDB/172(4)
BS2	SGAIB/204(13)	8/BP1	SGAIB/172(10)
BT2	SGCIB/210(13)	8/BM1	SGCIB/172(12)
BU2	$\overline{\text{CS3}}$ /219(4)	12/BU2	$\overline{\text{CS3}}$ /238(11)
BV2	$\overline{\text{RUN}}$ /228(3)	13/BV2	$\overline{\text{RUN}}$ /261(3)

Module 13. Control Panel Interface Board 2

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
AA1	MAR6/233(4)	8/AU2	MAR6/178(21)
AB1	MAR7/234(4)	8/AV2	MAR7/178(23)
AC1	MAR8/244(4)	8/BA1	MAR8/179(5)
AD1	MAR9/243(4)	8/BB1	MAR9/179(7)
AE1	MAR10/245(4)	8/BC1	MAR10/179(4)
AF1	MAR11/246(4)	8/BD1	MAR11/179(11)
AR1	MPC8/252(12)		
AS1	CONTO/261(12)		
AT1	GND		
AU1	CONT1/261(10)		
AV1	CONT2/261(8)		
AA2	Vcc		
AB2	CS4/238(9)	12/AB2	CS4/233(1)
AC2	GND		
AD2	AC0/256(1)	7/BB2	AC0/115(15)
AE2	AC1/256(3)	7/BD2	AC1/115(14)
AF2	AC2/256(5)	7/BE2	AC2/115(13)
AH2	AC3/256(13)	7/BF2	AC3/115(12)
AJ2	AC4/256(11)	7/BH2	AC4/116(15)
AK2	AC5/256(9)	7/BJ2	AC5/116(14)
AL2	AC6/257(1)	7/BK2	AC6/116(13)
AM2	AC7/257(3)	7/BL2	AC7/116(12)

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
AN2	AC8/257(5)	7/BM2	AC8/117(15)
AP2	AC9/257(13)	7/BN2	AC9/117(14)
AR2	AC10/257(11)	7/BP2	AC10/117(13)
AS2	AC11/257(9)	7/BR2	AC11/117(12)
AU2	LD/Cable 1-1	12/AU2	LD/228(8)
AV2	$\overline{MC}$ /251(13)	7/CF2	$\overline{MC}$ /142(3)
BA1	SR6/249(15)	4/BE1	SR6/60(5)
BB1	SR7/249(11)	4/BD1	SR7/60(2)
BC1	SR8/250(15)	4/BS1	SR8/59(11)
BD1	SR9/250(11)	4/BR1	SR9/59(8)
BE1	SR10/251(15)	4/BP1	SR10/59(5)
BF1	SR11/251(11)	4/BN1	SR11/59(2)
BH1	F8/252(11)	3/AM2	F8/50(11)
BJ1	F9/252(3)	3/AF2	F9/50(3)
BK1	F10/252(10)	3/AE2	F10/50(10)
BL1	F11/252(4)	3/AD2	F11/50(4)
BM1	MPC6/233(14)	12/BM1	MPC6/231(9)
BN1	MPC7/234(14)	12/BN1	MPC7/231(5)
BP1	NANO1/237(8)		
BR1	NANO2/238(8)		
BS1	BUS4/253(6)	12/BS1	BUS4/229(10)
BT1	GND		
BU1	BUS5/253(7)	12/BU1	BUS5/229(8)
BV1	IR4/253(10)		

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
BA2	Vcc		
BB2	IR5/253(9)		
BC2	GND		
BD2	$\overline{\text{BUS6}}/262(1)$	8/CF1	
BE2	$\overline{\text{BUS7}}/262(3)$	8/CE1	
BF2	$\overline{\text{BUS8}}/262(5)$	8/CD1	
BH2	$\overline{\text{BUS9}}/262(13)$	8/CC1	
BJ2	$\overline{\text{BUS10}}/262(11)$	8/CB1	
BK2	$\overline{\text{BUS11}}/262(9)$	8/CA1	
BL2	OSG2/249(1)	12/BL2	OSG2/228(10)
BM2	IRGB/247(13)	8/BK1	IRGB/172(6)
BP2	MBRSB/235(13)	8/BF1	MBRSB/172(2)
BR2	SGDB/241(13)	8/BH1	SGDB/172(4)
BS2	SGAB/236(13)	8/BP1	SGAIB/172(10)
BT2	SGCIB/242(13)	8/BM1	SGCIB/172(12)
BU2	$\overline{\text{CS3}}/238(11)$		

Module 16. Control State Generator, Run/Halt Flip-Flop

<u>Pin</u>	<u>Signal/IC</u>	<u>Module/Pin</u>	<u>Signal/IC</u>
AE1	CLK/367(2)	CLOCK	
AF1	$\overline{\text{STEP}}$ /--		
AH1	$\overline{\text{MC}}$ /366(12)	11/BB1	$\overline{\text{MC}}$
AJ1	HALTSTATE/365(2)		
AK1	SGA/365(4)	3/AS2	SGA/55(9)
AL1	SGB/365(8)		
AM1	SGC/366(2)	8/AL2	SGC/193(1)
AN1	SGD/366(6)	7/DS1	SGD/154(1)
AP1	SGE/366(8)	4/AP1	SGE/62(1)
AR1	SGF/366(10)	4/AS1	SGF/62(5)
AK2	SGAI/365(6)	1/AD2	SGAI/179(14)
AL2	SGBI/365(10)	11/BD2	SGBI
AM2	SGCI/366(4)	8/AK2	SGCI/182(14)
AS2	SGGI/365(12)	5/AS2	SGGI/79(14)

## Appendix E

### Microprogram Code Description

# Microprogram Code Description

Binary	Octal	Description
0 0 0 0 0 0	00	I/O interrupt
0 0 0 0 0 1	01	Evoke PC output
0 0 0 0 1 0	02	IF Link = 1, increment PC
0 0 0 0 1 1	03	Evoke A register output
0 0 0 1 0 0	04	Return to fetch
0 0 0 1 0 1	05	Memory cycle (Load)
0 0 0 1 1 0	06	IF Link = 0, increment PC
0 0 0 1 1 1	07	Evoke B register output
0 0 1 0 0 0	10	IF AC = 0, increment PC
0 0 1 0 0 1	11	Evoke A-1
0 0 1 0 1 0	12	IF AC $\neq$ 0, increment PC
0 0 1 0 1 1	13	Evoke memory output
0 0 1 1 0 0	14	IF ACo = 1, increment PC
0 0 1 1 0 1	15	Evoke MBR output
0 0 1 1 1 0	16	IF ACo = 0, increment PC
0 0 1 1 1 1	17	Evoke AC input
0 1 0 0 0 0	20	IF keyboard/reader flag = 1, increment PC
0 1 0 0 0 1	21	Evoke AC output
0 1 0 0 1 0	22	Evoke MAP output
0 1 0 0 1 1	23	IF printer/punch flag = 1, increment PC
0 1 0 1 0 0	24	Evoke Ad[MBR] output
0 1 0 1 0 1	25	IF printer/punch interrupt request flag = 1, or keyboard/reader interrupt request flag = 1 increment PC

# Microprogram Code Description

Binary	Octal	Description
0 1 0 1 1 0	26	Clear AC
0 1 0 1 1 1	27	Evoke MAR <sub>0-4</sub> output
0 1 1 0 0 0	30	Memory cycle (read)
0 1 1 0 0 1	31	Evoke SR output
0 1 1 0 1 0	32	Evoke transmitter buffer output
0 1 1 0 1 1	33	Increment PC
0 1 1 1 0 0	34	Evoke AdAC output
0 1 1 1 0 1	35	Evoke ADD
0 1 1 1 1 0	36	Evoke A+1
0 1 1 1 1 1	37	If IP <sub>4</sub> =0 clear MAR
1 0 0 0 0 0	40	If JMP, Evoke PC input
1 0 0 0 0 1	41	Evoke PC <sub>0-4</sub> input
1 0 0 0 1 0	42	Evoke AdPC input
1 0 0 0 1 1	43	Clear PC
1 0 0 1 0 0	44	Clear Link
1 0 0 1 0 1	45	Complement A
1 0 0 1 1 0	46	Complement Link
1 0 0 1 1 1	47	Evoke Receiver Buffer input
1 0 1 0 0 0	50	Shift AC right
1 0 1 0 0 1	51	Halt
1 0 1 0 1 0	52	Evoke inclusive OR
1 0 1 0 1 1	53	Evoke exclusive OR

# Microprogram Code Description

Binary	Octal	Description
1 0 1 1 0 0	54	Clear Keyboard/reader flag
1 0 1 1 0 1	55	Set Keyboard/reader flag
1 0 1 1 1 0	56	Evoke PC input
1 0 1 1 1 1	57	Set operate printer punch
1 1 0 0 0 0	60	Clear printer/punch flag
1 1 0 0 0 1	61	Set printer/punch flag
1 1 0 0 1 0	62	IF AC <sub>11</sub> = 1, set interrupt
1 1 0 0 1 1	63	Evoke A register input
1 1 0 1 0 0	64	Evoke B register input
1 1 0 1 0 1	65	Evoke MAR input
1 1 0 1 1 0	66	Evoke memory address input
1 1 0 1 1 1	67	Evoke MBR input
1 1 1 0 0 0	70	Evoke IR input
1 1 1 0 0 1	71	Shift AC left
1 1 1 0 1 0	72	Evoke AC input
1 1 1 0 1 1	73	Evoke MPC input
1 1 1 1 0 0	74	Evoke Ad MAR input
1 1 1 1 0 1	75	Memory deposit
1 1 1 1 1 0	76	Evoke AND
1 1 1 1 1 1	77	No operation

Appendix F

Pedagog II Instruction Set (Emulating PDP-8)

AD-A080 394 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL--ETC F/6 9/2

PEDAGOG II REALIZATION.(U)

DEC 79 J S SHEEHAN

UNCLASSIFIED AFIT/CE/EE/79-033

NL

3 of 3

ALC  
SUBROUTINE



END

DATE

FILED

3 - 80

DOC

## Memory Reference Instructions

<u>MNEMONIC</u>	<u>OCTAL</u>	<u>OPERATION</u>
AND	OXXX	<u>Logical AND</u> . The content of the memory location specified by XXX is combined with the content of the AC by a bit-wise logical AND operation. The result is left in the AC, the operand is restored to memory and the original content of the AC is lost.
TAD	1XXX	<u>Two's Complement Add</u> . The content of the memory location specified by XXX is combined with the content of the AC by two's complement addition. The result is left in the AC, the operand is restored to memory, and the original content of the AC is lost.
ISZ	2XXX	<u>Increment and Skip if Zero</u> . The content of the memory location specified by XXX is incremented by 1 and restored to memory. If the content of the referenced location becomes zero, the PC is incremented by 1 to skip the next sequential instruction. If the content of the referenced location does not become zero, the next instruction is executed.

## Memory Reference Instructions

<u>MNEMONIC</u>	<u>OCTAL</u>	<u>OPERATION</u>
DCA	3XXX	<u>Deposit and Clear the AC.</u> The content of the AC is stored in the memory location specified by XXX and the AC is set to zero. The original content of the referenced memory location is lost.
JMS	4XXX	<u>Jump to Subroutine.</u> The contents of the PC is stored in the memory location specified by XXX. The PC is then loaded with one more than the address of this location ( $XXX + 1$ ), so that the memory location following the referenced location is the next instruction to be executed. The content of the AC is not affected.
JMP	5XXX	<u>Jump.</u> The 12-bit address of the memory location specified by XXX is loaded into the PC, so that the instructions stored at this address will be the next instruction to be executed. The original content of the PC is lost. The content of the AC is not affected.

## Register Reference Instructions

<u>MNEMONIC</u>	<u>OCTAL</u>	<u>OPERATION</u>
NOP	7000	<u>No operation.</u> This instruction causes a delay in program execution.
IAC	7001	<u>Increment AC.</u> The content of the AC is incremented by 1.
RAL	7004	<u>Rotate AC left.</u> The content of AC 1-11 is shifted into AC 0-10. The content of AC is shifted into the Link, and the content of the Link shifted into AC 11.
RTL	7006	<u>Rotate two left.</u> Equivalent to two consecutive RAL operations.
RAR	7010	<u>Rotate AC right.</u> The content of AC 0-10 is shifted into AC 1-11. The content of the Link is shifted into AC; and the content of AC 11 is shifted into the Link.
RTR	7012	<u>Rotate two right.</u> Equivalent to two consecutive RAR operations.
CML	7020	<u>Complement Link.</u> The content of the Link is complemented.
CMA	7040	<u>Complement AC.</u> The content of each bit of the A is complemented.
CLL	7100	<u>Clear Link.</u> The Link is loaded with a binary 0.
CLA	7200	<u>Clear AC.</u> Each bit of the AC is loaded with a binary 0.

## Register Reference Instructions

<u>MNEMONIC</u>	<u>OCTAL</u>	<u>OPERATION</u>
HLT	7402	<u>Halt.</u> Clears the Run flip-flop so that program execution stops.
OSR	7404	<u>Logical OR with Switch Register.</u> The content of the programmer's console Switch Register (SR) is combined with the content of the AC by a bit wise logical OR operation. The result is left in the AC and the original content of the AC is lost. The content of the SR is not affected.
SKP	7410	<u>Skip.</u> The content of the PC is incremented by 1, to skip the next sequential instruction.
SNL	7420	<u>Skip on non-zero Link.</u> The content of the Link is sampled. If the Link contains a 1, the content of the PC is incremented to skip the next sequential instruction. If the Link contains a 0, the next instruction is executed.
SZL	7430	<u>Skip on zero Link.</u> The content of the Link is sampled. If the Link contains a 0, the content of the PC is incremented to skip the next sequential instruction. If the Link contains a 1, the next instruction is executed.

# Register Reference Instructions

<u>MNEMONIC</u>	<u>OCAL</u>	<u>OPERATION</u>
SZA	7440	<u>Skip on zero AC.</u> The content of each bit of the AC is sampled. If every bit contains a 0, the content of the PC is incremented to skip the next sequential instruction. If any bit contains a 1, the next instruction is executed.
SNA	7450	<u>Skip on non-zero AC.</u> The content of each bit of the AC is sampled. If any bit contains a 1, the content of the PC is incremented by 1 to skip the next sequential instruction. If every bit contains a 0, the next instruction is executed.
SMA	7500	<u>Skip on minus AC.</u> The content of AC is sampled. If AC contains a 1, indicating that the AC contains a negative two's complement number, the content of the PC is incremented to skip the next sequential instruction. If AC contains a 0, the next instruction is executed.
SPA	7510	<u>Skip on positive AC.</u> The content of AC is sampled. If AC contains a 0, indicating that the AC contains a positive two's complement number (or zero), the content of the PC is incremented to skip the next sequential instruction. If AC contains a 1, the next instruction is executed.

### Register Reference Instructions

<u>MNEMONIC</u>	<u>OCTAL</u>	<u>OPERATION</u>
XOR	7500	<u>Exclusive OR with Switch Register.</u> The contents of the SR is combined with the contents of the AC by a bit wise exclusive OR operation. The result is left in the AC and the original content of the AC is lost.

# Input/Output Transfer Instructions

<u>MNEMONIC</u>	<u>CODE</u>	<u>OPERATION</u>
KCF	6030	Clear keyboard/reader flag without operating the device.
KSF	6031	Skip the next instruction if the keyboard/reader flag is a 1.
KCC	6032	Clear the AC and the keyboard/reader flag.
KRS	6034	Read a character from the keyboard/reader buffer. The keyboard/reader flag is set when the operation is completed.
KIE	6035	Enable the keyboard/reader to cause program interrupts if AC bit 11 is a 1. Disable the keyboard/reader from causing interrupts if AC bit 11 is a 0.
TFL	6040	Set the printer/punch flag.
TSF	6041	Skip the next instruction if the printer/punch flag is a 1.
TCF	6042	Clear the printer/punch flag.
TPC	6044	Load the contents of AC bit positions 5-11 into the printer/punch buffer and operate the printer punch. The printer/punch flag is set when the operation is completed.
TSK	6045	Skip the next sequential instruction if either the printer/punch interrupt request flag or the keyboard/reader interrupt request flag is set.

Appendix G

Control Panel Connecting Cables

# Control Panel Connecting Cables

<u>Cable</u>	<u>Pins</u>	<u>Signal</u>
1	1	Link
	2-7	AC 11-6
	8-13	AC 5-0
	14	Vcc (to AC)
2	1	Vcc (to IR)
	2-7	Octal 11-6
	8-13	IR, 11-6
	14	Vcc (to Bus)
3	1	Gnd (to Sel Sw)
	2-9	Sel Sw 1-8
4	14	Vcc (to Octal)
	2-7	SR 11-6
5	14	Vcc (to SR)
	1	Gnd (to Con Sw)
6	2-7	PC 11-6
	14	Vcc (to Con Sw)
	2-7	Bus 11-6
	8-13	Bus 5-0
7	14	Vcc (to PC)
	1	Gnd (to inverter)
	2-7	GP 5-0
	8-13	IR 5-0
8	14	Vcc (to inverter)
	1	Gnd (to Octal)
	2-7	SR 5-0
9	8-12	Clock 1-5
	1	EXAMINE
	2-7	PC 5-0
	8	Gnd (to Lamp Test)
10	9	LOAD
	10	DEPOSIT
	14	Vcc (to Lamp Test)
	1-6	Micro States H-E
	7	Gnd
	8	HALT
	9	RUN
	10-11	Micro State F-G
	12	RunPB
	13	HltPB
	14	STEP
	15	MC
	16	Vcc

## Appendix H

### Unibus/Omnibus Bus Interface

## Unibus/Omnibus Bus Interface

### Introduction

Pedagog II has a 256 word, random access scratchpad memory. The memory is used for both microcode and the main program code. A larger non-volatile memory that would plug into the Unibus, would allow more sophisticated main programs to be run, expanding the capabilities of the Pedagog.

In a previous effort by Winfred G. Parris (Ref. 3), a Unibus/Omnibus interface circuit was designed to allow the addition of a 4096 word core memory. The design was not implemented. The interface was constructed and tested in this effort and the results are the topic of this appendix. Included in this section are: 1) a description of the DEC Omnibus, 2) a discussion of the bus interface design philosophy, and 3) a discussion of errors discovered in the interface design. The information presented here is to assist those who plan to complete the bus interface circuit.

### Omnibus

The DEC Omnibus (H919) (as used in the PDP-8) is a back-plane etched circuit board with ten H843 connectors mounted onto the board and wave soldered. The Omnibus is designed so that every pin in a given connector slot is defined (see Figure 15, page 188). All PDP-8 modules plug directly into any slot of the Omnibus.

PIN	A1	A2	B1	B2	C1	C2	D1	D2
A	TP	+5V	TP	+5V	TP	+5V	TP	+15V
B	TP	-15V	TP	-15V	TP	-15V	TP	-15V
C	SP GND *	GND	GND	GND	GND	GND	GND	GND
D	*MA0	EMA0	*MA4	INT STROBE	I/O PAUSE	*TP1	*MA8	*IR0
E	*MA1	EMA1	*MA5	BRK IN PROG	C0	*TP2	*MA9	*IR1
F	*GND	GND	*GND	GND	GND	*GND	*GND	*GND
H	*MA2	EMA2	*MA6	MA, MS LOAD CONT	C1	*TP3	*MA10	*IR2
J	*MA3	*MEM START	*MA7	OVERFLOW	C2	*TP4	*MA11	*F
K	*MD0	*MD DIR	*MD4	BREAK DATA CONT	BUS STROBE	*TS1	*MD8	*D
L	*MD1	SOURCE	*MD5	BREAK CYCLE	INTERNAL I/O	*TS2	*MD9	*E
M	*MD2	*STROBE	*MD6	LD AND ENABLE	NOT LAST XFER	*TS3	*MD10	USER MODE
N	*GND	*GND	*GND	GND	GND	*GND	*GND	GND
P	*MD3	INHIBIT	*MD7	INT IN PROG	INT RST	*TS4	*MD11	F SET
R	DATA0	RETURN	DATA 4	RES 1	INITIALIZE	LINK DATA	DATA8	LOAD ADDRESS
S	DATA1	MEMORY READ	DATA 5	RES 2	SKIP	LINK LOAD	DATA 9	*STOP
T	GND	GND	GND	GND	GND	GND	GND	*GND
U	DATA 2	ROM ADDRESS	DATA 6	RUN	CPMA DISABLE	IND 1	DATA 10	KEY CONTROL
V	DATA 3	*LINK	DATA 7	POWER OK	MS, IR DISABLE	IND 2	DATA 11	SW

\* THIS PIN IS CONNECTED TO GROUND ON THE BUS, BUT SERVES AS A LOGIC SIGNAL WITHIN MODULES TO FACILITATE TESTING

Fig. 15. Omnibus Pin Assignments

Figure 21, page 190, shows the proposed additions to Pedagog. To add the core memory and I/O Controller (each of which plugs into the Omnibus) seven card slots in the Omnibus would be required. The circuit cards are:

- 4096 word, 12 bits/word core memory (MM8-E)
- X/Y Current Control for core memory (G227)
- Sense/Inhibit circuitry for core memory (G104)
- I/O Controller (M8650)
- Timing Generator (M8330)
- Bus Terminator (M8320)
- Bus Interface circuit card (W966)

One additional card would have to be plugged into the Unibus side of the backplane. It is the bus interface card for the Unibus (W968) (see Figure 16, page 191).

A portion of the Pedagog backplane is constructed to act as an Omnibus, and provides eight card slots. The left quarter of the Pedagog backplane (as seen from the rear) is used as an Omnibus. Bussing strips connect the appropriate pins together as the wave soldering does on the DEC Omnibus (H919) module. The Omnibus requires three power supplies: +15 VDC, -15 VDC, and +5 VDC. These power supplies are available in the Pedagog cabinet. Figure 15, page 188, indicates which of the Omnibus pins connect to the power supplies. Note that four rows of pins in each quadrant of the backplane connect to ground. These pins have been prewired to each other but must be tied to the system ground.

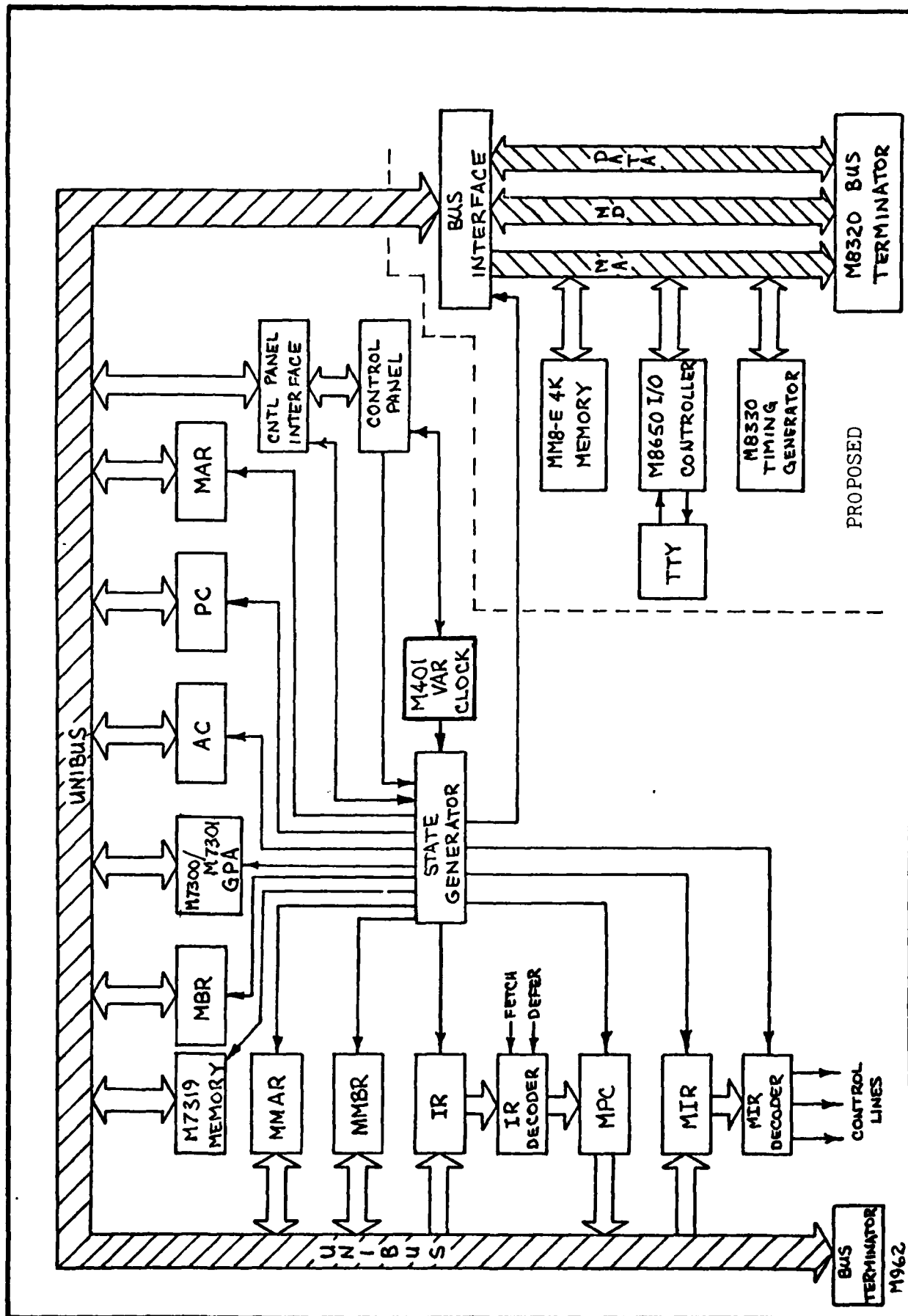


Fig. 21. Pedagog Block Diagram

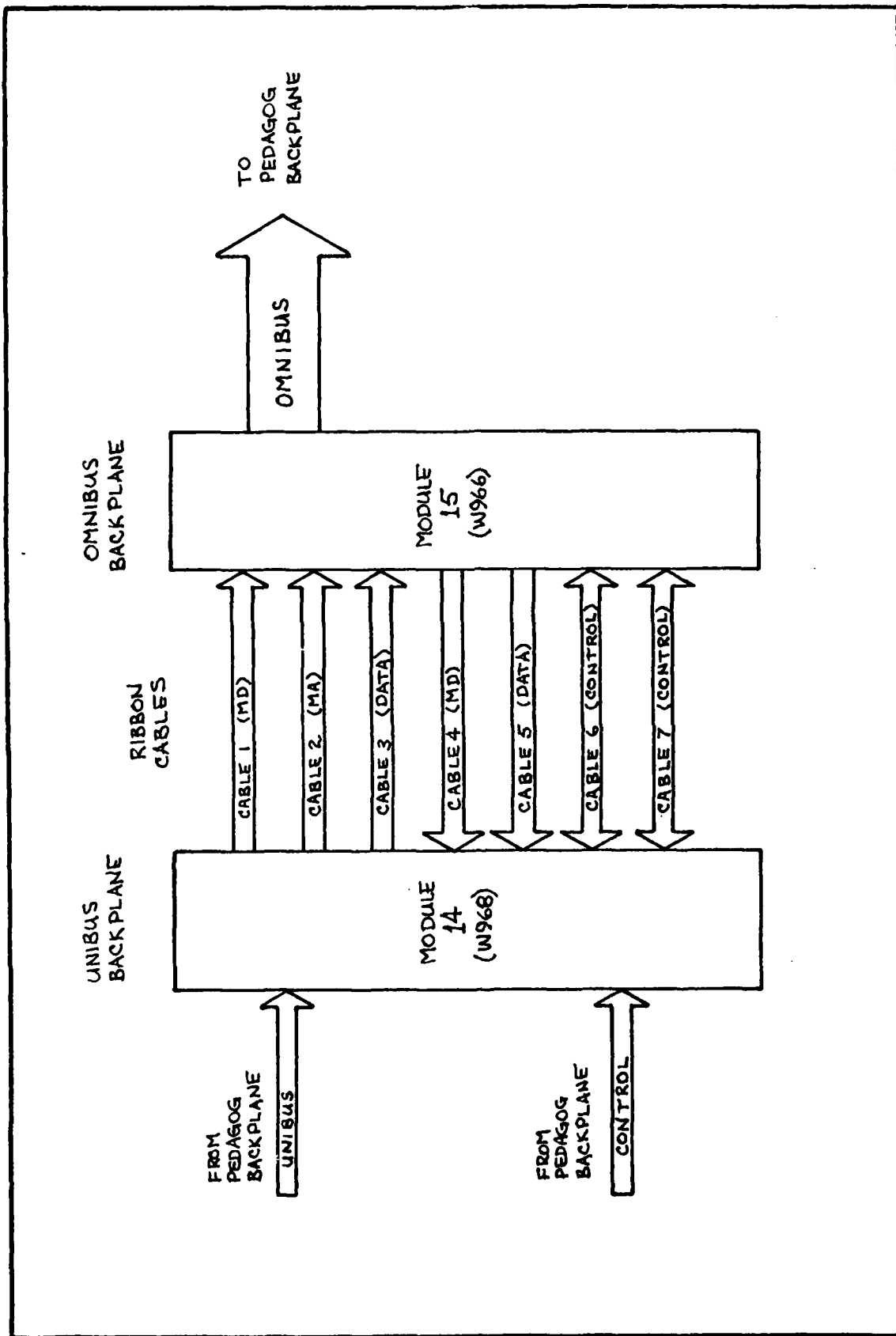


Fig. 16. Bus Interface Block Diagram

Three separate busses make up the Omnibus. They are the Memory Address (MA) bus, Memory Data (MD) bus, and the DATA bus. The MA bus is used to address memory in any one of 4096 possible locations. The bus is 12 bits wide and is uni-directional. The MD bus has 12 memory data lines that carry information to and from memory. The direction of data flow on these lines is controlled by a signal called MD DIR (see Figure 15, pin AK2, page 188). If MD DIR is held low, information stored in the sense flip-flops of the currently active memory is placed on the MD lines. When MD DIR is high, the contents of the MD register in the bus interface are placed on the MD lines. As in the case of the MA lines, the MD lines are low for a one and high for a zero. The DATA bus is 12 bits wide and serves as a bi-directional multi-purpose bus. Generally, these lines are used as input/output lines. For Pedagog's purposes, not all of the Omnibus signals need to be used. However, since PDP-8 RTM's, which insert into the Omnibus, sense most of these unused signals, attention must be given to properly terminating even the unused signals. Several of these signals will be discussed in detail in this appendix in the section on design errors. A complete description of all Omnibus signals is available in Ref. 7:12-21.

#### Interface Design Philosophy

The Omnibus/Unibus interface designed by Parris (Ref. 3: 29-42) was not a total bus interface. Since only two sections

of the Omnibus (the MA and MD busses) were necessary to read/write memory, only the MA and MD busses were designed into the interface. Parris' design left the control signals to interface the DATA bus unspecified (see Figures 62, page 196, and 64, page 198). To be able to plug other Omnibus compatible modules into the Omnibus (such as an I/O controller, M8650) the DATA bus must be totally interfaced to the Unibus.

Parris' interface design was organized around the use of 12-bit latches (four 7474 IC's) to save the information being transferred, and open collector AND gates (7400) to feed the information onto the appropriate bus (see Figures 60 through 64, starting page 194). Information going from the Unibus to the Omnibus was stored using Pedagog timing signals. Two separate registers stored the information and transferred it to the Omnibus via open-collector AND gates. Control transfer signals on the Omnibus side of the interface were generated by a DEC Timing Generator Module (M8330).

The Timing Generator module is used in the PDP-8E to provide the signals to control memory and the central processor. The module generates a series of timing pulses (TP) and timing states (TS) which act as control signals. There are four timing states, labeled TS1, TS2, TS3, and TS4 (see Figure 15, column C2, page 188). These signals are used throughout the interface control circuits to sequence operations (see Figure 65, page 199).

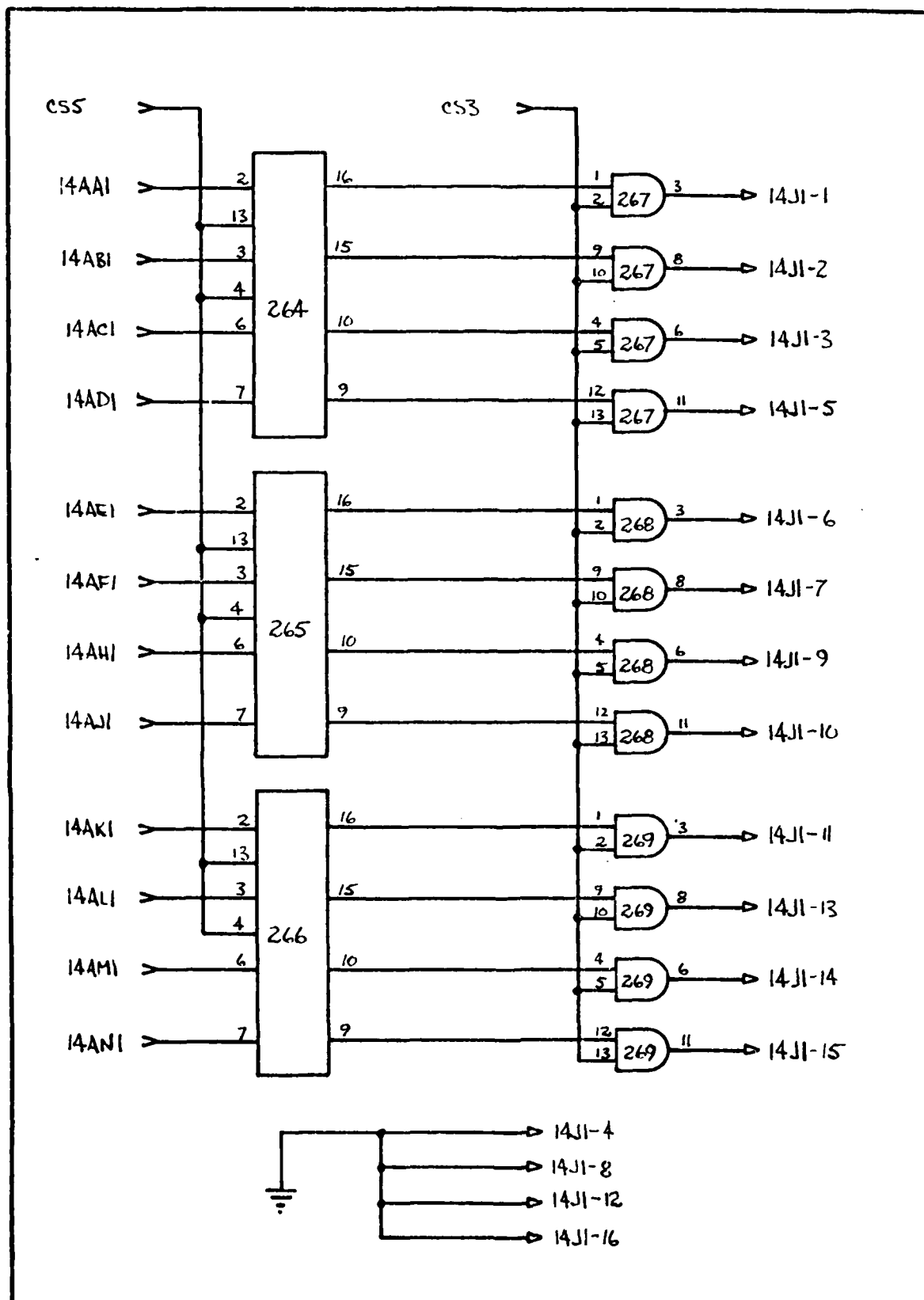


Fig. 60. Unibus/Omnibus Memory Data (MD) Interface

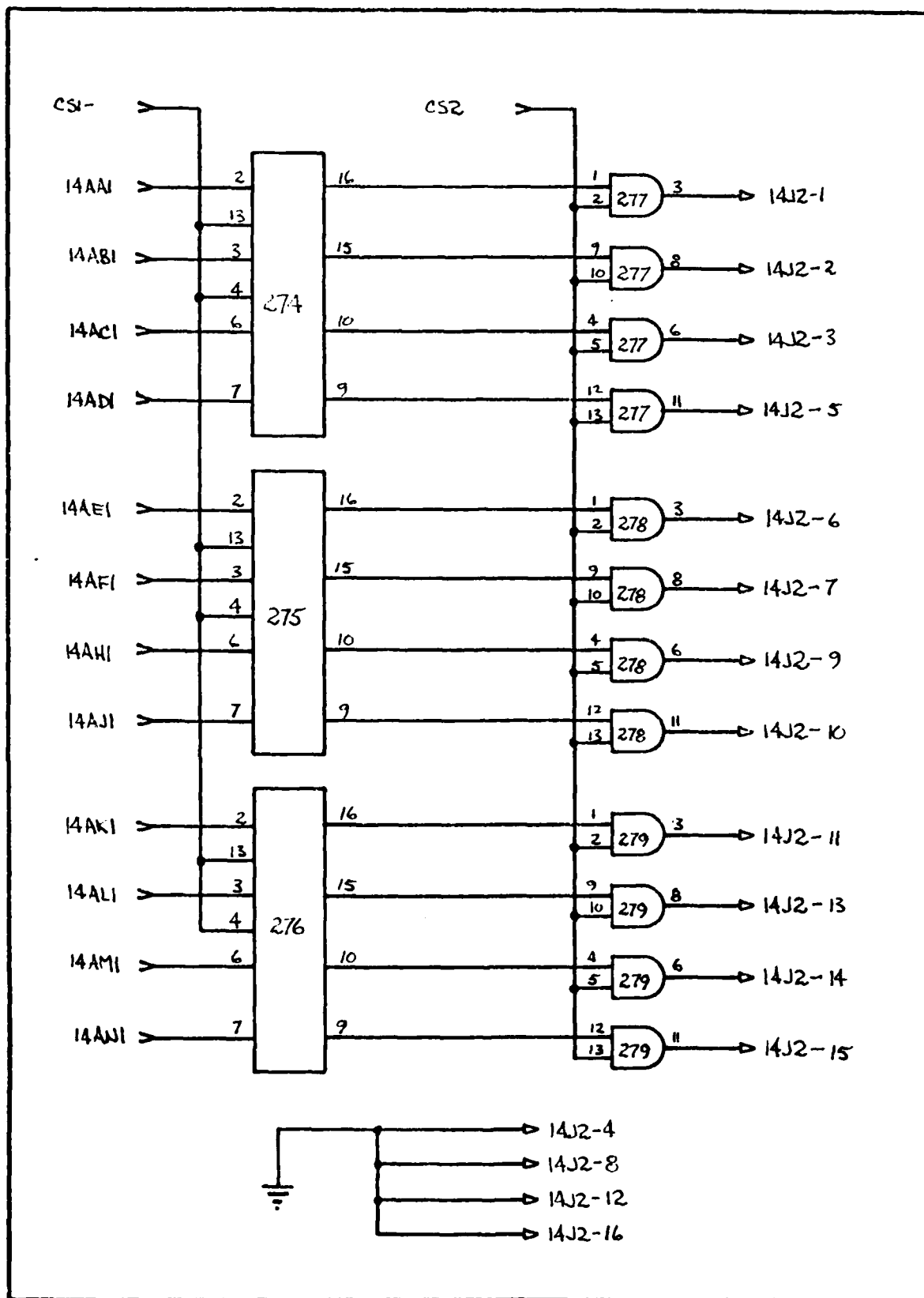


Fig. 61. Unibus/Omnibus Memory Address (MA) Interface

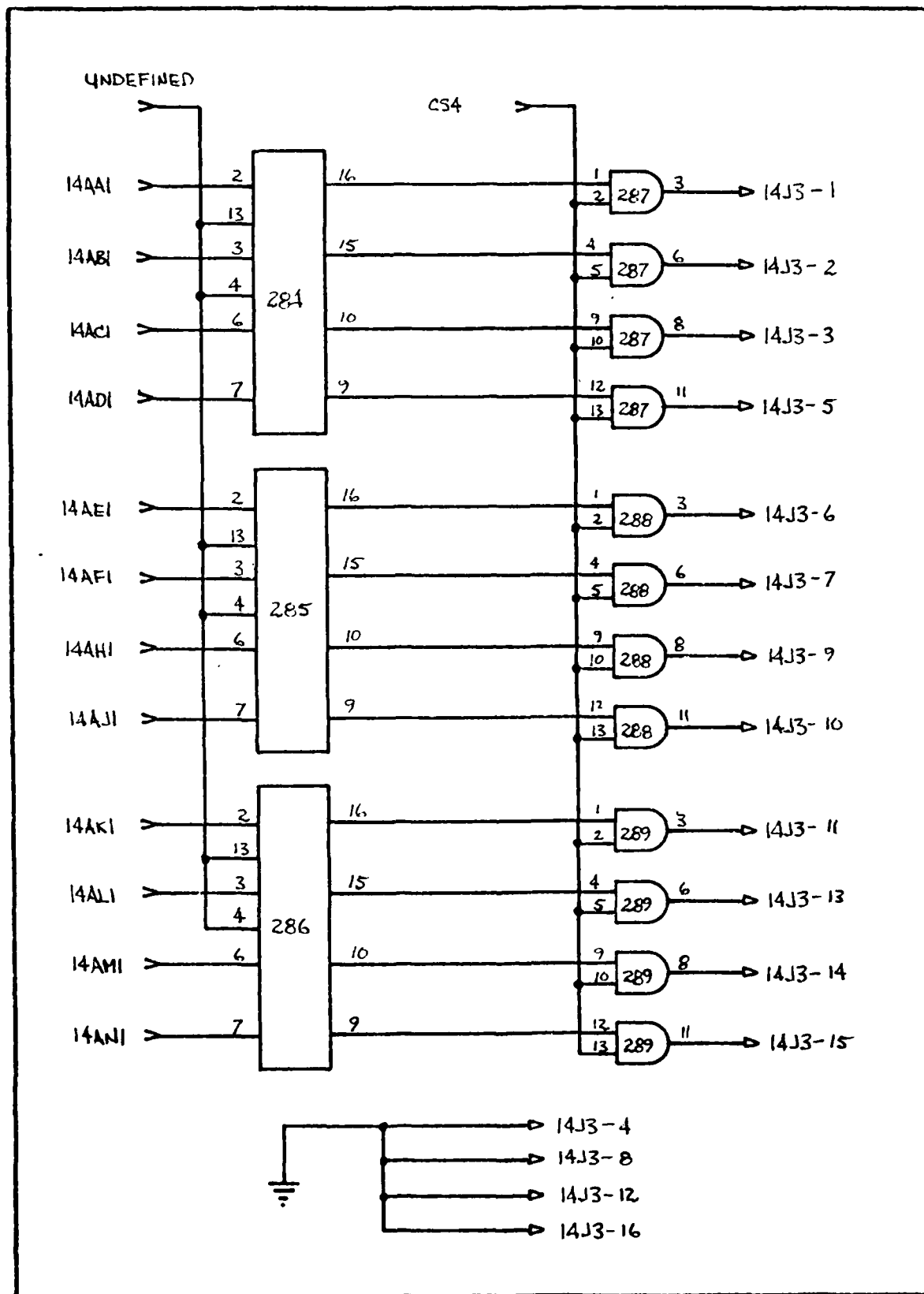


Fig. 62. Unibus/Omnibus Data Interface

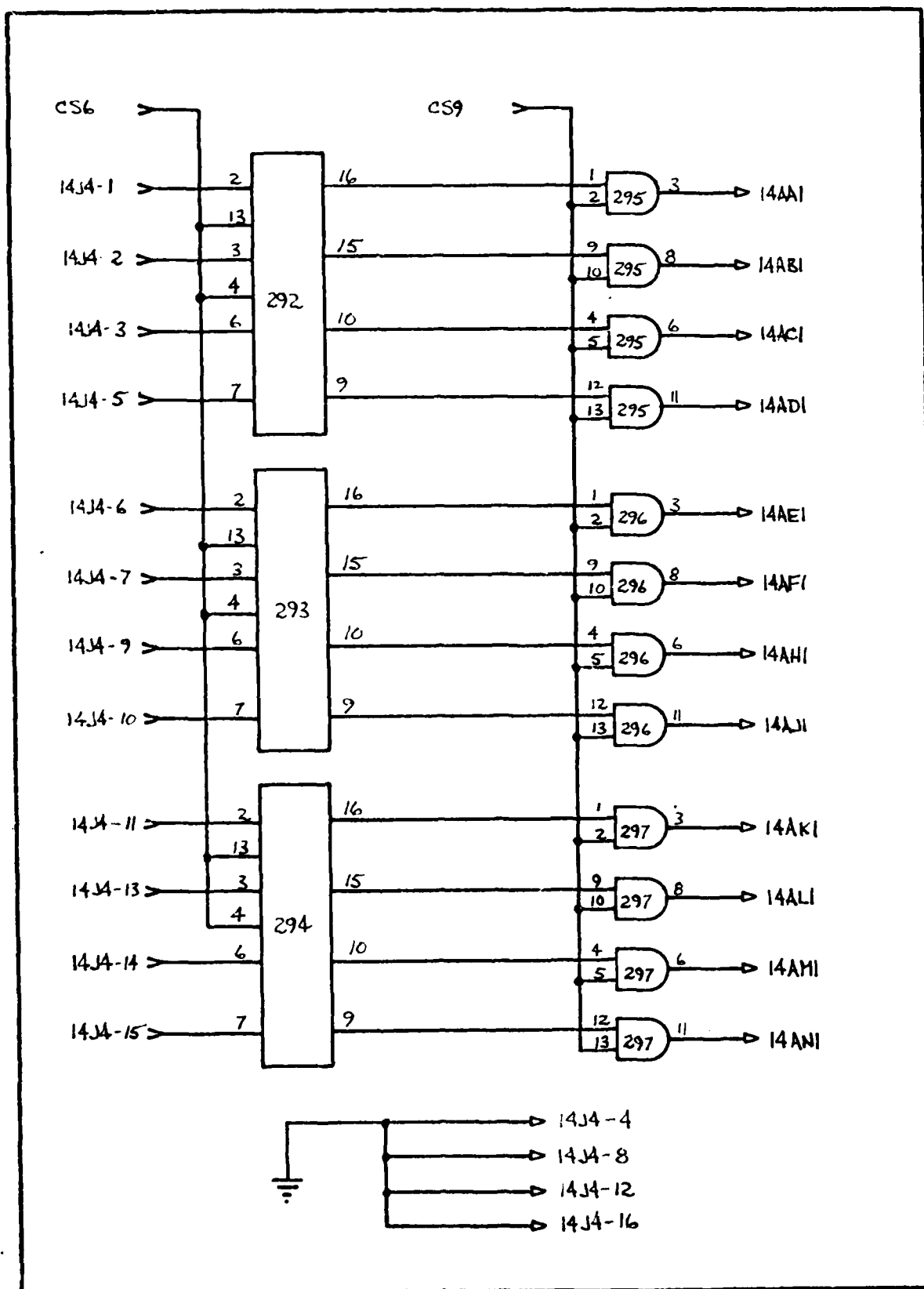


Fig. 63. Omnibus/Unibus Memory Data (MD) Interface

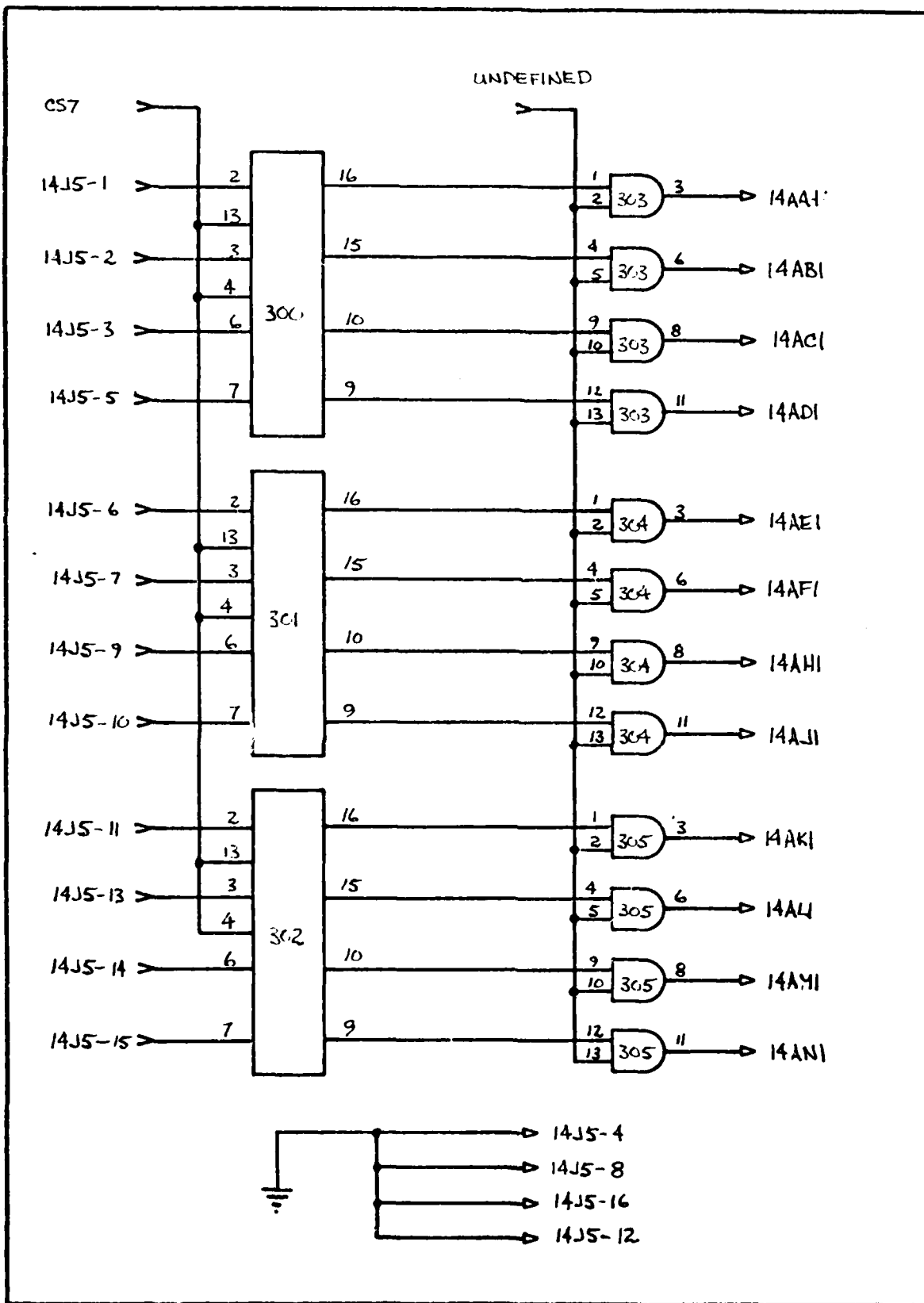


Fig. 67. Omnibus/Unibus Data Interface

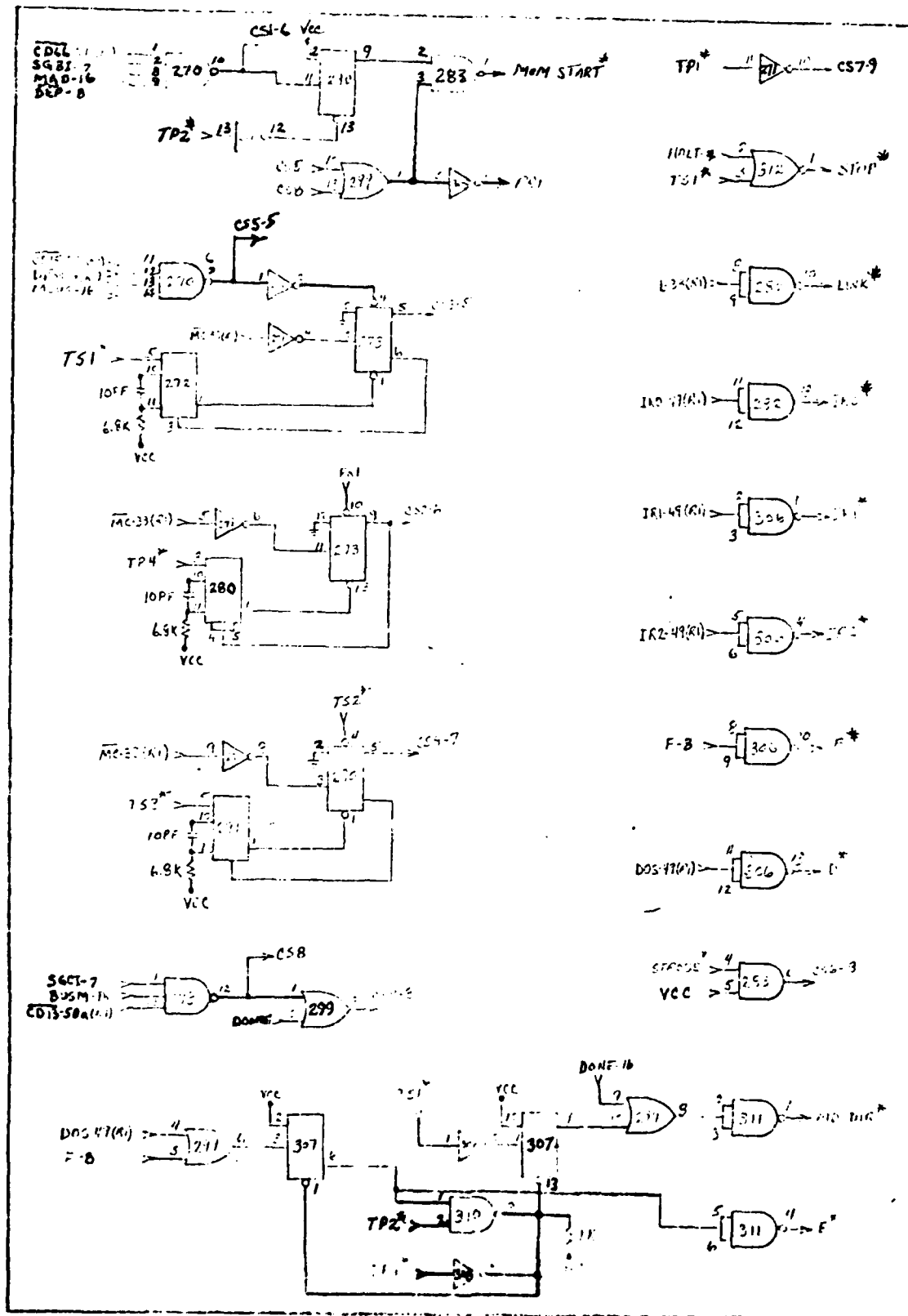


Fig. 65. Bus Interface Control Circuits

Memory data information being transferred from the Omnibus to the Unibus was collected from the MD bus. Open-collector AND gates were used to gate the register information onto the Unibus.

The bus interface circuitry is located on two blank DEC modules (W968 and W969). The two modules were designed to be connected with 16-conductor ribbon cables. The cables were designated C1 through C10 by Parris, and are associated with J1 (jack) through J10 (see Figures 67, page 201, and 68, page 202).

In this effort the bus interface circuit was temporarily constructed on SK-10 solderless breadboards. Only cables C1 through C5 were actually used in the interface construction. Cables C6-C10 are available if the interface is expanded to include the DATA bus. The circuit was tested and the results are discussed in the next section.

#### Design Errors

When the bus interface circuits were constructed in this analysis, several design errors (or omissions) were noted. Figures 60 through 64, starting page 194, show the interface transfer registers with their associated open-collector AND gates. The Omnibus and Unibus both use a high voltage level as a logic 0, and a low voltage level as a logic 1. If the interface circuit is constructed as designed by Parris, the open-collector AND gates which feed each bus will always leave the bus (both Unibus and Omnibus) in an inverted state. For example,

# Omnibus/Unibus Interface

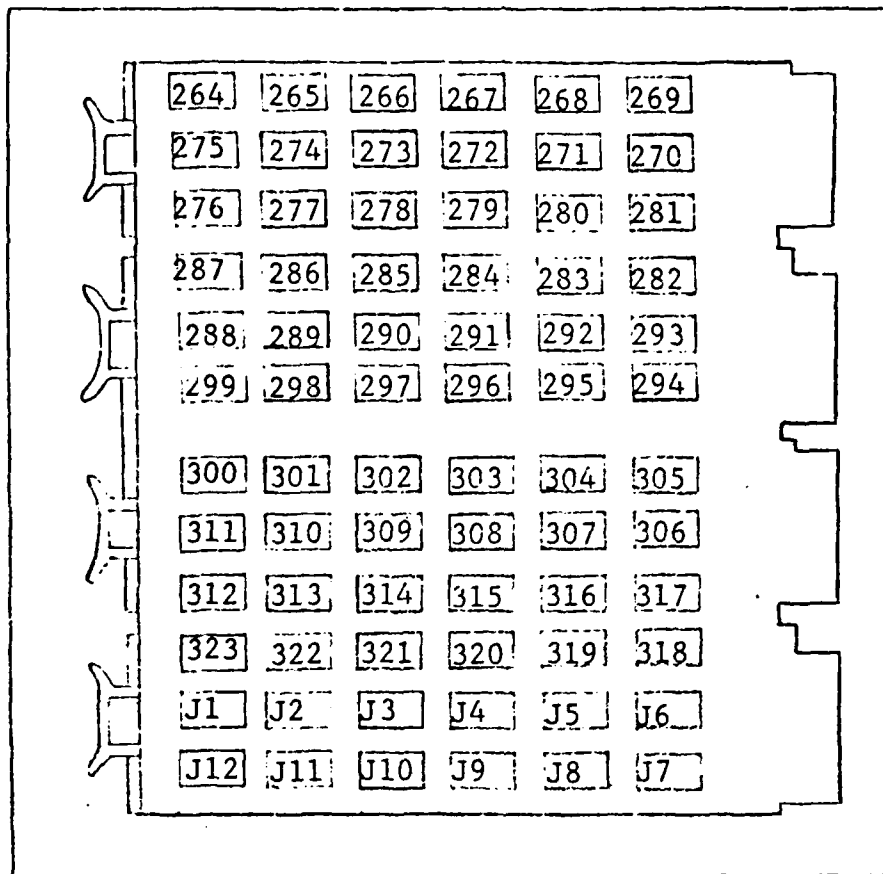


Fig. 67 W968 (Module 14) Layout

<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>
264	SN7475	274	SN7475	284	SN7475
265	SN7475	275	SN7475	285	SN7475
266	SN7475	276	SN7475	286	SN7475
267	SN7409	277	SN7409	287	SN7409
268	SN7409	278	SN7409	288	SN7409
269	SN7409	279	SN7409	289	SN7409
270	SN7420	280	SN74121	290	SN7474
271	SN7404	281	SN74121	291	SN74121
272	SN74121	282	SN7401	292	SN7475
273	SN7474	283	SN7408	293	SN7475

# Omnibus/Unibus Interface

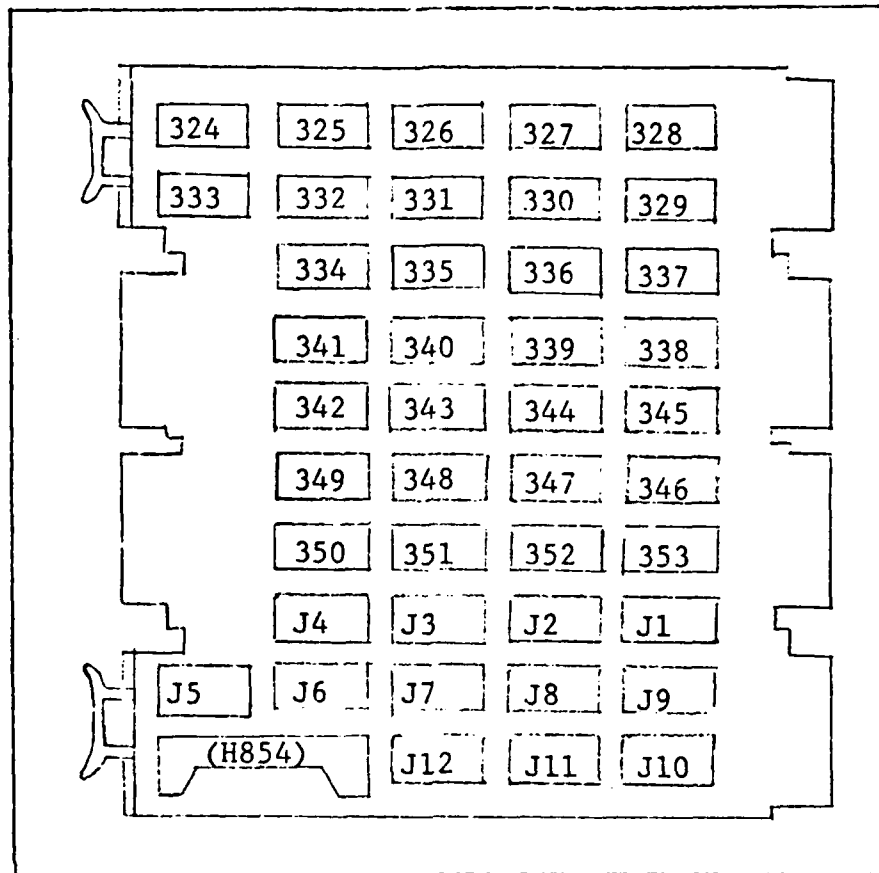


Fig. 68 W966 (Module 15) Layout

<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>	<u>Position</u>	<u>Type</u>
324		334		344	
325		335		345	
326		336		346	
327		337		347	
328		338		348	
329		339		349	
330		340		350	
331		341		351	
332		342		352	
333		343		353	

in Figure 61, page 195, the control signal CS2 is normally low (cleared to a low condition by  $\overline{MC}$ ). This causes the output of each gate to be at a logical 0, regardless of the input from its associated transfer latch. Since the bus (either Omnibus or Unibus) interprets a low state as a logical 1, the octal display will always show 1's when it should show 0's. This is exactly what occurred during testing of the interface circuit. The problem can be corrected by replacing each of the open-collector AND gates in Figures 60 through 64 with open-collector NAND gates.

The second design error noted was in the interface control logic (see Figure 65, page 199). Note that the bottom illustration in the figure makes improper use of the wired-and circuit. The NAND gate (310) and the inverter (308) have their outputs tied together with a pull-up resistor to Vcc. The circuit will not operate properly in this configuration. The problem can be corrected by either replacing the two gates with open-collector types, or inserting a totem-pole output AND gate at the junction of the NAND gate and inverter. The latter action was taken (see Figure 69, page 204).

The next two errors discovered in the interface design were errors of omission. The Omnibus has several pins which must be properly terminated even though they are not used. The POWER OK pin (Figure 15, BV2, page 188), when high, indicates to

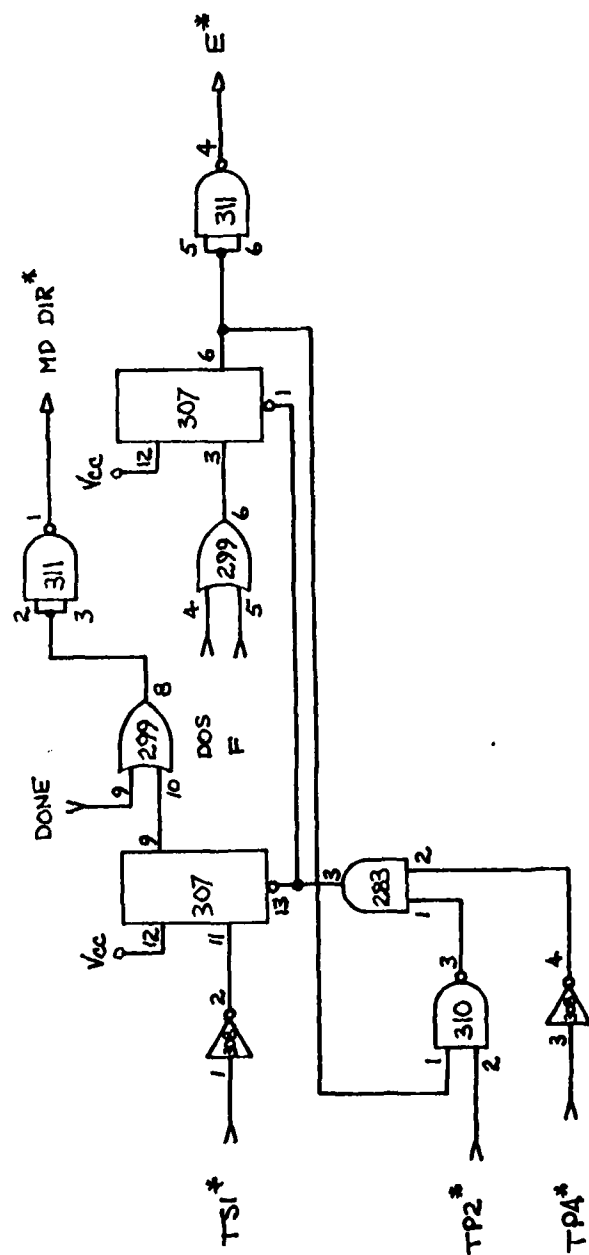


Fig. 69. Corrected Interface Control Circuit

the Timing Generator that the DC voltage from the power supply of the PDP-8 is adequate to allow proper functioning of the machine. If this line becomes negated, no new memory cycles will be started. The Timing Generator module will not operate with POWER OK negated. The signal must be tied to the +5 volt supply through a pull-up resistor.

The KEY CONTROL pin of the Omnibus (Figure 15, DU2, page 188) has a complicated function when used in the PDP-8 (Ref. 7:19). KEY CONTROL should be tied high during normal operation. After power-up, it must be taken to a logic 0 and returned to a logic 1 before the Timing Generator will operate. When the KEY CONTROL and POWER OK signals were corrected, the Timing Generator properly developed the control signals TP1 through TP4, and TS1 through TS4 (Ref. 8:3-55).

Due to time constraints, a total analysis of the bus interface circuit could not be completed. The next section discusses recommendations for future effort on the Pedagog bus interface.

#### Recommendations

The Unibus/Omnibus interface design proposed by Parris (Ref. 3:29-39) should be expanded. The bus interface should be extended to include the DATA bus as well as the MD and MA busses. If the DATA bus were a part of the interface, the Omnibus would be completely compatible with any DCE PDP-8 module. This would allow the 4096 word memory and one additional module to be added

to the Pedagog by just plugging them into the Omnibus. (The maximum number of slots in the Pedagog Omnibus is seven). The control signals for the DATA bus are complicated and will require considerably more control circuitry than that designed by Parris (see Figure 65, page 199).

It should be noted that the control circuitry presented by Parris in Figure 65 does have design errors. If a complete bus interface design is attempted, the circuitry in Figure 65 should probably be discarded and the new interface design be done from the top down. This would prevent redundancy and unnecessary complications.

An overall bus interface design would need to include several control signals from Pedagog that are not used in the present bus interface design. These signals were provided in the original design of Pedagog in the event an I/O controller was added to the system. The signals are available at the output of the MIRD and are listed in Table I.

Four control signals in Figure 37, page 100 would also have to be a part of the interface design. The signals are KRF, PPF, PRIRF, and KRIRF. Note that each of these signals has an unspecified origin. This is because the signals originate outside the Pedagog. The signals are:

KRF	Keyboard/reader flag
PPF	Printer/punch flag

Table I

## Pedagog Input/Output Control Signals

<u>Signal</u>	<u>Function</u>
CD00	Input/output interrupt
CD20	If keyboard/reader flag = 1, increment PC
CD23	If printer/punch flag = 1, increment PC
CD25	If printer/punch interrupt request flag = 1, or keyboard/reader interrupt request flag = 1, increment PC
CD32	Evoke transmitter buffer output
CD47	Evoke receiver buffer input
CD54	Clear keyboard/reader flag
CD55	Set keyboard/reader flag
CD57	Set operate printer punch
CD60	Clear printer/punch flag
CD61	Set printer/punch flag
CD62	If $AC_{11} = 1$ , set interrupt

PRIRF      Printer/punch interrupt request flag

KRIRF      Keyboard/reader interrupt request flag

The signals listed and those used by Parris (Figure 65, page 199) would have to be incorporated into a control circuit to govern the transfer of data across the bus interface. The number of signals involved, both those listed here and those on the Omnibus would make the design of a complete bus interface a challenging project.

Appendix J

Register Transfer Language Description

## Register Transfer Language Symbols

Symbol	Symbol Description
[ ]	The square brackets denote contents of a register. Thus [IP] means the contents of the IP. [IRi] denotes the contents of the ith bit of the IP. Ad [IP] means the address part of the contents of the IP.
M<MAR>	This notation denotes the location of a memory word addressed by the MAR.
[M<MAR>]	This notation denotes the contents of the memory word location addressed by the MAR.
→	The arrow means the transfer of information from one register to another or from one register bit to another. Thus, [AC] → IP denotes that the contents of the AC are transferred to the IP register. [AC] → M<MAR> means that the contents of the AC are transferred to the memory location addressed by the MAR. [Ai-1] → Ai means that the contents of the (i-1) bit of the AC is transferred to the ith bit of the AC. 1 → AC means that a 1 is transferred into the AC.
+	This symbol has two distinct meanings depending upon whether Boolean variables are involved or addition is indicated. If Boolean variables are involved, a logical OR is indicated. If total register contents are indicated an arithmetic addition using two's complement arithmetic is indicated. If this symbol is utilized with individual register bit contents (Boolean variables), it will mean a logical OR.

## Symbol

## Symbol Description

: A colon following a Boolean variable denotes the occurrence of the following Boolean statement when the value of the variable is 1. This particular variable is usually a control signal generated from the decoder. Thus,  $q_{5t1} : [AC] \rightarrow MBR$  means that when the value of  $q_{5t1}$  is 1, the contents of the AC are transferred to the MBR.

## Boolean Operations

The Boolean operations are standard. That is, a AND operation can be denoted as  $AB$ ,  $A \cdot B$ ,  $(A)(B)$ , or  $(A) \cdot (B)$ . The OR operation is denoted by the  $A + B$  or  $(A) + (B)$ . The exclusive OR operation is denoted by  $A \oplus B$ .

IF The IF is used as a logical condition. When the condition listed after IF is true, the register transfer occurs. For example, the PTL IF  $IR_4 = 0$ ,  $0 \rightarrow MAR$ . When this instruction is encountered, the MAR will be cleared if  $IR_4$  is a zero.

## Register Transfer Language Description

### Master State Generator

[MPC]  $\rightarrow$  MMAR

[MMAR]  $\rightarrow$  Mad, [M<MMAR>]  $\rightarrow$  MMBP

[MMBR]  $\rightarrow$  [IR], [MPC] + 1  $\rightarrow$  MPC

[MIRD]  $\rightarrow$  R    R = register

### Fetch

[PC]  $\rightarrow$  MAP

[MAR]  $\rightarrow$  Mad, [M<MAR>]  $\rightarrow$  MBR

[MBR]  $\rightarrow$  IR

[PC] + 1  $\rightarrow$  PC, [IRD]  $\rightarrow$  MPC

### Indirect Addressing

IF IR<sub>4</sub> = 0, 0  $\rightarrow$  MAR

Ad [MBR]  $\rightarrow$  Ad MAR

[MAR]  $\rightarrow$  Mad, [M<MAR>]  $\rightarrow$  MBR

[PC] + 1  $\rightarrow$  PC

IF [IR<sub>0-2</sub>] = 101, MBR  $\rightarrow$  PC

[IRD]  $\rightarrow$  MPC

### Memory Reference Instructions

#### AND

Ad [MBR]  $\rightarrow$  Ad MAR

[MAR]  $\rightarrow$  Mad, [M<MAR>]  $\rightarrow$  MBR

[MBR]  $\rightarrow$  A

[AC]  $\rightarrow$  B

[A]  $\cdot$  [B]  $\rightarrow$  AC

7500  $\rightarrow$  MPC

### TAD

Ad [MBR] → Ad MAR

[MAR] → Mad, [M<MAR>] → MBR

[MBR] → A

[AC] → B

[A] + [B] → AC

7500 → MPC

### DSZ

[AC] → B

Ad [MBR] → Ad MAR

[MAR] → Mad, [M<MAR>] → A

[A] -1 → AC

IF [AC] = 0, [PC] + 1 → PC

[AC] → MBR

[MAR] → Mad, [MBR] → M<MAR>

B → AC

7500 → MPC

### JMP

Ad [MBR] → AdPC

IF IR<sub>4</sub> = 1, [MAR<sub>0-4</sub>] → PC<sub>0-4</sub>

7500 → MPC

### JMS

Ad [MBR] → Ad MAR

[PC] → MBR

[MAR] → Mad, [MBR] → M<MAR>

[MAR] → PC

[PC] + 1 → PC

7500 → MPC

### DCA

Ad [MBR] → Ad MAR

[AC] → MBR

[MAR] → Mad, [MBR] → M<MAR>

0 → AC

7500 → MPC

### Register Reference Instructions

#### CLA

0 → AC , 7500 → MPC

#### CLL

0 → L , 7500 → MPC

#### CMA

[AC] → A

$\overline{[A]}$  → AC

7500 → MPC

#### CML

[L] → L , 7500 → MPC

#### IAC

[AC] → A

[A] + 1 → AC

7500 → MPC

RAR

$[AC_i] \rightarrow AC_{i+1}$  ,  $i = 0, \dots, 10$ ,  $[AC_{11}] \rightarrow L$

$[L] \rightarrow AC$  ,  $7500 \rightarrow MPC$

RTR

$[AC_i] \rightarrow AC_{i+1}$  ,  $i = 0, \dots, 10$ ,  $[AC_{11}] \rightarrow L$

$[L] \rightarrow AC$

$[AC_i] \rightarrow AC_{i+1}$  ,  $i = 0, \dots, 10$ ,  $[AC_{11}] \rightarrow L$

$[L] \rightarrow AC$  ,  $7500 \rightarrow MPC$

RAL

$[AC_i] \rightarrow AC_{i+1}$  ,  $i = 11, \dots, 1$ ,  $[AC_0] \rightarrow L$

$[L] \rightarrow AC_{11}$  ,  $7500 \rightarrow MPC$

RTL

$[AC_i] \rightarrow AC_{i-1}$  ,  $i = 11, \dots, 1$ ,  $[AC_0] \rightarrow L$

$[L] \rightarrow AC_{11}$

$[AC_i] \rightarrow AC_{i-1}$  ,  $i = 11, \dots, 1$  ,  $[AC_0] \rightarrow L$

$[L] \rightarrow AC_{11}$  ,  $7500 \rightarrow MPC$

NOP

$7500 \rightarrow MPC$

HALT

$0 \rightarrow RUN$ ,  $7500 \rightarrow MPC$

OSR

$[AC] \rightarrow A$

$[SR] \rightarrow B$

$[A] + [B] \rightarrow AC$

$7500 \rightarrow MPC$

SKP

$[PC] + 1 \rightarrow PC$ ,  $7500 \rightarrow MPC$

SNL

IF [L] = 1, [PC] + 1 → PC

7500 → MPC

SZL

IF [L] = 0, [PC] + 1 → PC

7500 → PMC

SZA

IF [AC] = 0, [PC] + 1 → PC

7500 → MPC

SNA

IF [AC] = 0, [PC] + 1 → PC

7500 → MPC

SMA

IF [AC] = 1, [PC] + 1 → PC

7500 → MPC

SPA

IF [AC] = 0, [PC] + 1 → PC

7500 → MPC

XOR

[AC] → A

[SR] → B

[A] ⊕ [B] → AC

7500 → MPC

Input/Output Transfer Instructions

KCF

0 → KRF

7500 → MPC

KSF

IF KRF = 1, [PC] + 1 → PC

7500 → MPC

KCC

0 → AC, 0 → KPF

7500 → MPC

KRS

Ad [MBR] → Ad MAR

[TB] → MBR

[MAR] → Mad, [MBR] → M<MAR>

1 → KRF, 7500 → MPC

KIE

IF [AC<sub>11</sub>] = 1, 1 → INT

7500 → MPC

TFL

1 → PPF, 7500 → MPC

TSF

IF PPF = 1, [PC] + 1 → PC

7500 → MPC

TCF

0 → PPF, 7500 → MPC

TPC

Ad [AC] → RB

1 → Operate Printer/Punch

1 → PPF, 7500 → MPC

TSK

( IF PPIRF = 1 or KRIRF = 1, [PC] + 1 → PC

7500 → MPC

Appendix K

Pedagog II User's Manual

## Pedagog II User's Manual

### Introduction

This appendix contains the operating instructions for Pedagog II, a general-purpose, microprogrammable computer. Pedagog is unique, in that it allows the user to watch actual register transfers take place inside the machine.

This manual is written specifically for the user. Little knowledge of the internal workings of this machine are necessary to program and operate it. However, abbreviations of register names are used and can be found in Table II. It is assumed that very little is known about the concept of microprogramming. Therefore, this manual takes a step-by-step approach to programming and operating the Pedagog.

The user should note that Pedagog has a random access memory. When power to the machine is turned off, the contents of memory are lost and must be re-entered when the machine is turned on again. For this reason, most users will find that working sessions of at least one hour are the most productive. During user absences (to go to class, for example) it does not harm the machine to leave it running.

### Power Up

The power switch is located on the right side of the table top. Insure that the machine is plugged into a wall outlet. Press the POWER ON switch. The cooling fans inside the cabinet will come on along with a light in the POWER ON switch. To turn

Table II  
Register Abbreviations

<u>Abbreviation</u>	<u>Register</u>
AC	Accumulator
GPA	General Purpose Arithmetic Unit
IR	Instruction Register
IRD	Instruction Register Decoder
L	Link
MAR	Memory Address Register
MBR	Memory Buffer Register
MIR	Microprogram Instruction Register
MIRD	Microprogram Instruction Register Decoder
MMAR	Microprogram Memory Address Register
MMBR	Microprogram Memory Buffer Register
PC	Program Counter

the power off, simply press the POWER OFF switch. An OVERTEMP light is located above the POWER ON/OFF switch. If this light comes on, the machine should shut down automatically. If not, manually turn the power off and call a technician.

When the power is turned on, the panel will light up randomly. Locate the CLR (clear) control switch in the center of the panel (see Figure 18, page 222). Push the switch down. This will clear the bus and set selected internal registers to zero in preparation for machine operation. Note that the CLR switch, as well as all other control switches, have only a DOWN position. Do not push these switches up. If these switches are pushed to the UP position, they could break off. The machine is now ready for operation. The next section will give a brief description of the panel and the function of each of its switches.

#### Panel Description

Octal Display. Located in the upper center of the panel is the OCTAL DISPLAY. Each digit in the display represents three binary bits of a 12-bit binary word. The greatest decimal number that can be represented by three binary bits is seven. Therefore, the largest number that can be displayed by the OCTAL DISPLAY is 7777. The OCTAL DISPLAY can be used to show the contents of the bus, or the contents of seven different registers. The registers are listed directly under the OCTAL DISPLAY in what is called the DISPLAY INDICATOR.

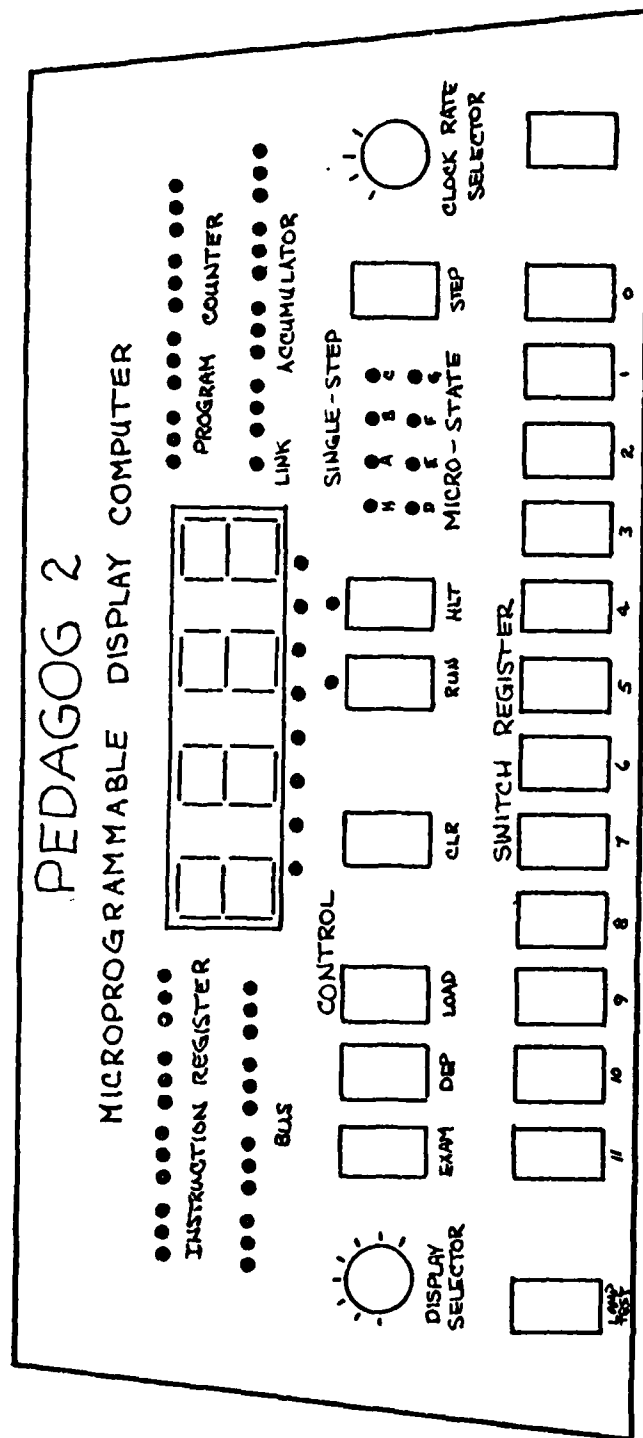


Fig. 18. Control Panel

Display Indicator. The DISPLAY INDICATOR is a series of eight lamps that indicate which register is being displayed in the OCTAL DISPLAY. Only one display indicator lamp will be lit at one time.

Display Selector. The DISPLAY SELECTOR is a rotary switch on the left center of the panel. It has eight positions and is used to select a particular register for display in the OCTAL DISPLAY.

Fixed Displays. Four FIXED DISPLAYS are located in the upper left and right of the panel. Each display always shows the contents of one register and one register only. Each display is made up of 12 lamps set in groups of three for easy visual conversion to octal. Note that the ACCUMULATOR DISPLAY has associated with it a LINK bit. The other fixed displays are the BUS DISPLAY, INSTRUCTION REGISTER DISPLAY, and PROGRAM COUNTER DISPLAY.

Control Switches. Seven control switches are located in the center of the panel. They are: 1) EXAM MEM (examine memory), 2) DEP MEM (deposit in memory), 3) LOAD ADDR (load address), 4) CLR (clear), 5) RUN (run), 6) HLT (halt), and 7) STEP (step). The function of each switch is discussed below:

a. EXAM MEM--examines the memory location specified by the Program Counter. The switch must be activated twice. The first activation displays the contents of the memory location, and the second activation clears the display from the bus.

b. DEP MEM--stores the contents of the switch register into the memory location indicated by the Program Counter.

c. LOAD ADDR--allows the address in the Switch Register to be loaded into the Program Counter.

d. CLR--sets the GPA, accumulator, and Link to a binary zero, sets the state generator to the halt state, and sets the MPC to address 7400 in memory. Memory is also cleared of any pending read/write operation.

e. RUN--activates the system to full operation. The Pedagog will run only from the Halt state.

f. HLT--stops system operation. The Pedagog can also be halted under software control by using the HALT command. This command will be discussed later.

Switch Register. The SWITCH REGISTER is located along the bottom of the panel. It is used to enter data and addresses into the machine.

Clock Rate Selector. The CLOCK RATE SELECTOR determines how fast the machine will operate in the RUN mode. The machine operates fastest on the 100 Khz setting, but any clock rate can be used. The slower speeds are used primarily for hardware debugging. The CLOCK RATE SELECTOR has no effect when using the single-step mode (the single-step mode is discussed later).

Micro State Indicators. Eight MICRO STATE INDICATORS are located in the right center of the panel. These indicators show which of eight states the machine is in.

Lamp Test. The LAMP TEST switch lights all segments of the OCTAL DISPLAY to insure all segments operate. The switch is spring-loaded and will return to the center position after being pushed down.

#### Memory Layout

Pedagog has a very small memory that is shared between microcode and main program code. The memory is a 256 word random access memory. This section describes the memory and discusses what parts are used for microcode, and what parts are available for the main program.

Pedagog microprogramming is set up on a memory mapped basis. This means that specific locations of memory must be used to store microcode because the computer always goes to the same location in memory to refer to the same piece of microcode. Not only does this mean that the microcode must be in a specific location, but it also means that any program error which causes one of these locations to be changed will cause the machine to give unpredictable results.

The memory has  $400_8$  12-bit storage locations. Only eight bits are needed to address  $400_8$  locations ( $2^8 = 256_{10} = 400_8$ ). Therefore, when entering addresses through the SWITCH REGISTER on the panel, only the least significant eight bits need to be used. When the 4096 word memory is added, the locations from  $7400_8$  to  $7777_8$  will contain the machine microcode. This is

because the clear switch sets the MPC to 7400, the address of the first word of microcode. To be consistent, this manual will use all 12 bits, and will treat the 400<sub>8</sub> memory locations as those from 7400<sub>8</sub> to 7777<sub>8</sub>. Just keep in mind that 7400<sub>8</sub> (111100000000) is the same address as 0000<sub>8</sub> (000000000000). That is, the least significant eight bits are the same.

One hundred and fifty words (base eight) of microcode are stored beginning at location 7500 in memory. (Henceforth, all numbers will be in base eight unless the subscript indicates otherwise.) One word is stored at location 7400. This leaves locations 7401 to 7477, and 7651 to 7777 for main program storage (see Table III). In the next section, the microcode that emulates the PDP-8 will be entered into memory.

#### Entering Microcode

Pedagog II was designed to emulate the PDP-8. Pedagog's 12-bit word and register architecture were patterned after the 12-bit word and architecture of the PDP-8. The microcode in Table III allows the PDP-8 instruction set to be programmed into Pedagog. Even though Pedagog was designed to emulate the PDP-8, many different instruction sets could be programmed into Pedagog. This section describes the steps necessary to micro-program Pedagog to execute the 36 basic instructions of the PDP-8.

Table III  
Microprogramming to Emulate the PDP-8

<u>Command</u>	<u>Location</u>	<u>Contents</u>
	7400	0477
	Available for Main Program	
Fetch	7500	0165
	7501	3067
	7502	1570
	7503	3373
Defer	7504	3777
	7505	2474
	7506	3067
	7507	3377
	7510	1556
	7511	7773
AND	7512	2474
	7513	3067
	7514	1563
	7515	2164
	7516	1776
	7517	0477
	7520	----
	7521	----
TAD	7522	2474
	7523	3067
	7524	1563
	7525	2164
	7526	3572
	7527	0477
	7530	----
	7531	----
	7532	----
ISZ	7533	2164
	7534	2474
	7535	3063
	7536	3672
	7537	1077

Table III (Cont)

<u>Command</u>	<u>Location</u>	<u>Contents</u>
	7540	2167
	7541	0577
	7542	0772
	7543	0477
JMP	7544	2442
	7545	2741
	7546	0477
	7547	----
JMS	7550	2474
	7551	0167
	7552	0577
	7553	2256
	7554	3377
	7555	0477
	7556	----
	7557	----
DCA	7560	2474
	7561	2167
	7562	0577
	7563	2677
	7564	0477
	7565	----
CLA	7566	2677
	7567	0477
CLL	7570	0444
	7571	----
CMA	7572	2163
	7573	1745
	7574	0477
	7575	----
CML	7576	0446
	7577	----
IAC	7600	2163
	7601	3672
	7602	0477
	7603	----

Table III (Cont)

<u>Command</u>	<u>Location</u>	<u>Contents</u>
RAR	7604	0450
	7605	----
RTR	7606	7750
	7607	0450
	7610	----
RAL	7611	0471
	7612	----
RTL	7613	7771
	7614	0471
	7615	----
NOP	7616	0477
	7617	----
HALT	7620	7751
	7621	----
OSR	7622	2163
	7623	3164
	7624	1752
	7625	0477
	7626	----
SKP	7627	3377
	7630	0477
SNL	7631	0277
	7632	0477
SZL	7633	0677
	7634	0477
SZA	7635	1077
	7636	0477
SNA	7637	1277
	7640	0477
SMA	7641	1477
	7642	0477
SPA	7643	1677
	7644	0477

Table III (Cont)

<u>Command</u>	<u>Location</u>	<u>Contents</u>
XOR	7645	2163
	7646	3164
	7647	1753
	7650	0477
<div style="border: 1px solid black; padding: 5px; display: inline-block;">                     Available for Main Program                 </div>		

The Pedagog microcode is listed in Table III. To deposit a microword into memory, follow the steps listed below:

Set the DISPLAY SELECTOR to the BUS position

Press the CLR switch (Sets MPC to 7400 and clears machine)

Set the SWITCH REGISTER to 7400 (up is a 1, down is a 0)

Press the LOAD ADDR switch (To load the starting address)

Set the SWITCH REGISTER to 0477 (First microword)

Press the DEP MEM switch (Loads first microword)

The above steps have entered the microword 0477 into memory location 7400. To examine the location 7400 in memory, follow the steps listed below:

Set 7400 in the SWITCH REGISTER (Address of microword)

Press the LOAD ADDR switch (Load address of microword)

Press the EXAM MEM switch (Examine contents of 7400)

The contents of memory location 7400 will appear on the OCTAL DISPLAY. The EXAM MEM switch must be pressed a second time to clear the data from the bus. If the data is not cleared from the bus, the next time a register output is enabled to the bus, it will be superimposed on what is being put on the bus by memory. This will cause erroneous results. The OCTAL DISPLAY should now show zero (0000). Note that as a location is deposited or examined, the Program Counter is incremented. This allows sequential memory locations to be filled (or examined) without having to set a new address each time.

The complete list of microcode in Table III should be entered into memory by using the same procedure as the above example. In the locations where no microcode is listed (like 7520 and 7521) any value can be entered.

#### Microcode Description

Each four-digit word of microcode that was entered into memory in the previous section is actually two micro instructions. The first two digits represent the first instruction, and the last two digits represent the second instruction. For example, the first four-digit word listed in Table III, 0477, is actually the two instructions 04 (return to fetch), and 77 (no operation). When a four-digit word is decoded, two instructions are being carried out at one time. This is especially convenient when a register transfer needs to be made. For example, the second four-digit word in Table III, 0165, enables the Program Counter output (01), and enables the Memory Address Register input (65). In this way, the contents of the Program Counter are transferred to the MAR by executing just one microword.

The microprogram code description for each instruction is listed in Table IV. The word "evoke" is often used in place of enable. Register transfer abbreviations in the table are described in Appendix J. Note that the microprogram code description table is in two sections (octal codes 00 through 37 and codes 40 through 77). The machine implementation requires that codes 00 through 37 be used only in the first two positions of the

Table IV

## Microprogram Code Description

Binary	Octal	Description
0 0 0 0 0 0	00	I/O interrupt
0 0 0 0 0 1	01	Evoke PC output
0 0 0 0 1 0	02	IF Link = 1, increment PC
0 0 0 0 1 1	03	Evoke A register output
0 0 0 1 0 0	04	Return to fetch
0 0 0 1 0 1	05	Memory cycle (Load)
0 0 0 1 1 0	06	IF Link = 0, increment PC
0 0 0 1 1 1	07	Evoke B register output
0 0 1 0 0 0	10	IF AC = 0, increment PC
0 0 1 0 0 1	11	Evoke A-1
0 0 1 0 1 0	12	IF AC $\neq$ 0, increment PC
0 0 1 0 1 1	13	Evoke memory output
0 0 1 1 0 0	14	IF ACo = 1, increment PC
0 0 1 1 0 1	15	Evoke MBR output
0 0 1 1 1 0	16	IF ACo = 0, increment PC
0 0 1 1 1 1	17	Evoke AC input
0 1 0 0 0 0	20	IF keyboard/reader flag = 1, increment PC
0 1 0 0 0 1	21	Evoke AC output
0 1 0 0 1 0	22	Evoke MAP output
0 1 0 0 1 1	23	IF printer/punch flag = 1, increment PC
0 1 0 1 0 0	24	Evoke Ad[MBR] output
0 1 0 1 0 1	25	IF printer/punch interrupt request flag = 1, or keyboard/reader interrupt request flag = 1 increment PC

Table IV (Cont)  
Microprogram Code Description

Binary	Octal	Description
0 1 0 1 1 0	26	Clear AC
0 1 0 1 1 1	27	Evoke MAR <sub>0-4</sub> output
0 1 1 0 0 0	30	Memory cycle (read)
0 1 1 0 0 1	31	Evoke SR output
0 1 1 0 1 0	32	Evoke transmitter buffer output
0 1 1 0 1 1	33	Increment PC
0 1 1 1 0 0	34	Evoke AdAC output
0 1 1 1 0 1	35	Evoke ADD
0 1 1 1 1 0	36	Evoke A+1
0 1 1 1 1 1	37	IF IP <sub>4</sub> =0 clear MAR
1 0 0 0 0 0	40	IF JMP, Evoke PC input
1 0 0 0 0 1	41	Evoke PC <sub>0-4</sub> input
1 0 0 0 1 0	42	Evoke AdPC input
1 0 0 0 1 1	43	Clear PC
1 0 0 1 0 0	44	Clear Link
1 0 0 1 0 1	45	Complement A
1 0 0 1 1 0	46	Complement Link
1 0 0 1 1 1	47	Evoke Receiver Buffer input
1 0 1 0 0 0	50	Shift AC right
1 0 1 0 0 1	51	Halt
1 0 1 0 1 0	52	Evoke inclusive OR
1 0 1 0 1 1	53	Evoke exclusive OR

Table IV (Cont)  
Microprogram Code Description

Binary	Octal	Description
1 0 1 1 0 0	54	Clear Keyboard/reader flag
1 0 1 1 0 1	55	Set Keyboard/reader flag
1 0 1 1 1 0	56	Evoke PC input
1 0 1 1 1 1	57	Set operate printer punch
1 1 0 0 0 0	60	Clear printer/punch flag
1 1 0 0 0 1	61	Set printer/punch flag
1 1 0 0 1 0	62	IF $AC_{11} = 1$ , set interrupt
1 1 0 0 1 1	63	Evoke A register input
1 1 0 1 0 0	64	Evoke B register input
1 1 0 1 0 1	65	Evoke MAR input
1 1 0 1 1 0	66	Evoke memory address input
1 1 0 1 1 1	67	Evoke MBR input
1 1 1 0 0 0	70	Evoke IR input
1 1 1 0 0 1	71	Shift AC left
1 1 1 0 1 0	72	Evoke AC input
1 1 1 0 1 1	73	Evoke MPC input
1 1 1 1 0 0	74	Evoke Ad MAR input
1 1 1 1 0 1	75	Memory deposit
1 1 1 1 1 0	76	Evoke AND
1 1 1 1 1 1	77	No operation

four-digit word, and that the codes 40 through 77 be used only in the last two positions. For example, the four-digit code 6501 (in location 7500) would be incorrect because the first code (65) is greater than 37, and the last code (01) is less than 40. The one exception to this rule is the "no operation" code (77), which can appear in either position (for example, location 7511 in Table II is 7773).

Using the microprogram code descriptions listed in Table IV, a large number of microprogram instructions are possible. It is this flexibility in defining various instructions that makes a microprogrammable computer a flexible tool. The next two sections deal with main level program instructions and how they are defined using micro instructions.

#### Main Program Instructions

Main program instructions are 12 bits long, the same as the microcode word length. There are three types of words used at the main program level: instructions, addresses, and data. Although all have the same length, each has a completely different function.

The instruction set used by the programmer is the same as the DEC PDP-8 instruction set. The advantage of using the PDP-8 instruction set is that programs written for the PDP-8 can be run on the Pedagog. Note, however, that programs using DEC microprogramming (the ability to execute two PDP-8 instructions as one) cannot be run. Only those programs using, or

adapted to, the PDP-8 basic instruction set (36 instructions) are allowed.

The instruction word format for program instructions consist of memory reference instructions (MRI), register reference instructions (RRI), and input/output instructions (I/O). The I/O capability of Pedagog has not yet been incorporated, so that instruction group is not discussed here (see Ref. 1:19-20).

The word format used for an MRI is shown in Figure 70, page 240. Note that the bit numbering starts at the left with 0 and goes to the right to 11. DEC uses this convention in its PDP-8 literature. (Other systems may use bit numbering conventions which start at the right with 0 and go to the left to 11.) All references to bit positions in this manual will conform to the DEC convention, that is, 0 is the leftmost (most significant) bit and 11 is the rightmost bit. Note, however, that the Pedagog SWITCH REGISTER is numbered from left (11) to right (0), and care must be taken not to confuse bit numbering.

The first three bits (0-2) of the MRI specify the operation code. For the MRI, this will be an octal number of 0 through 5, as indicated in the instruction set shown in Table V. Bit 3 is the indirect address bit. A zero indicates direct memory access and the data is in the memory location specified by the address bits 5-11. A one in bit 3 indicates an indirect memory access. In this case, the address bits 5-11 indicate the memory

Table V

## Memory Reference Instructions

<u>MNEMONIC</u>	<u>OCTAL</u>	<u>OPERATION</u>
AND	0XXX	<u>Logical AND.</u> The content of the memory location specified by XXX is combined with the content of the AC by a bit-wise logical AND operation. The result is left in the AC, the operand is restored to memory and the original content of the AC is lost.
TAD	1XXX	<u>Two's Complement Add.</u> The content of the memory location specified by XXX is combined with the content of the AC by two's complement addition. The result is left in the AC, the operand is restored to memory, and the original content of the AC is lost.
ISZ	2XXX	<u>Increment and Skip if Zero.</u> The content of the memory location specified by XXX is incremented by 1 and restored to memory. If the content of the referenced location becomes zero, the PC is incremented by 1 to skip the next sequential instruction. If the content of the referenced location does not become zero, the next instruction is executed.

Table V (Cont)  
Memory Reference Instructions

<u>MNEMONIC</u>	<u>OCTAL</u>	<u>OPERATION</u>
DCA	3XXX	<u>Deposit and Clear the AC.</u> The content of the AC is stored in the memory location specified by XXX and the AC is set to zero. The original content of the referenced memory location is lost.
JMS	4XXX	<u>Jump to Subroutine.</u> The contents of the PC is stored in the memory location specified by XXX. The PC is then loaded with one more than the address of this location ( $XXX + 1$ ), so that the memory location following the referenced location is the next instruction to be executed. The content of the AC is not affected.
JMP	5XXX	<u>Jump.</u> The 12-bit address of the memory location specified by XXX is loaded into the PC, so that the instructions stored at this address will be the next instruction to be executed. The original content of the PC is lost. The content of the AC is not affected.

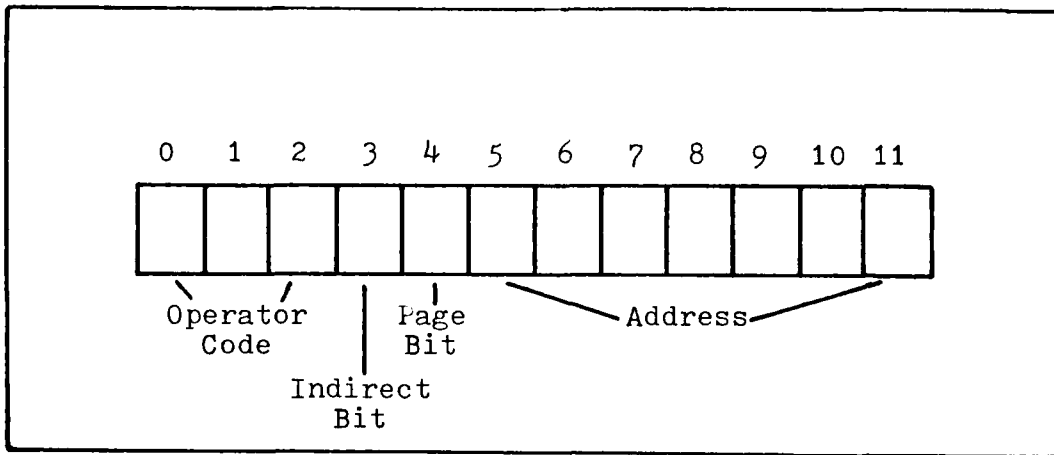


Fig. 70. MRI Word Format

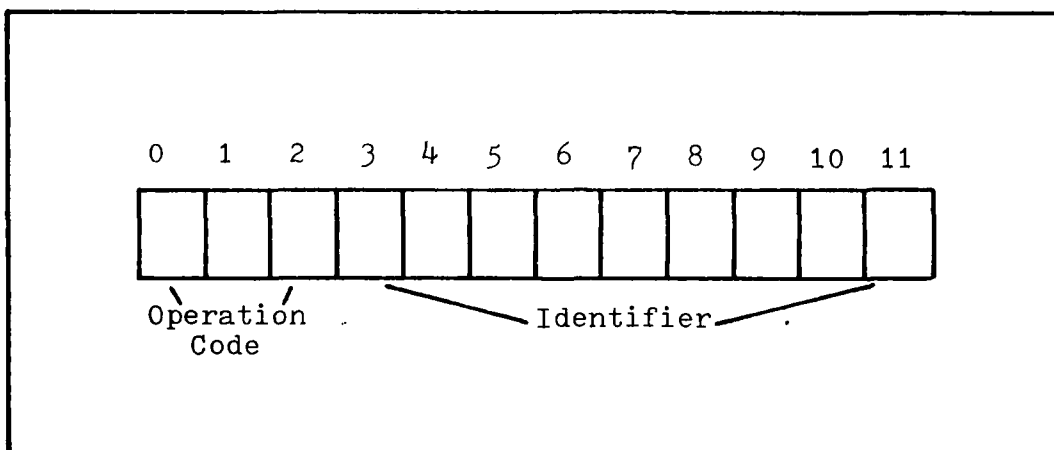


Fig. 71. RRI Word Format

location where the actual data address is located. Using the direct address, only the lower  $200_8$  locations can be addressed. Using indirect, a full 12 bit address can be specified and all  $7777_8$  locations can be addressed ( $2^{12} = 7777_8$ ). Only one level of indirect addressing is incorporated into Pedagog.

Bit four of the MRI instruction word is the paging bit. (PDP-8 paging and indirect addressing are described in detail in Ref. 9:2-13.) If bit four is zero, the zero page in memory is indicated (locations  $0-177_8$ ). If bit four is a one, the current page is indicated. For example, the instruction "AND 50" with bit four as a zero, indicates the data is in location fifty on the zero page. That is, location 50. The same instruction with bit four set to one indicates location 50 on the current page.

The word format for an RRI instruction is shown in Figure 71, page 240. Bits 0-2 again give the operation code. A seven indicates an RRI. Bits 3-11 identify the specific RRI to be accomplished. Register reference instructions are shown in Table VI. The next section describes how microcode is used to make each RRI or MRI occur.

#### Microcoding Main Program Instructions

This section describes how a main program instruction is made up of microprogram instructions. As an example, an instruction is taken from Table II and listed below:

Table VI

## Register Reference Instructions

<u>MNEMONIC</u>	<u>OCTAL</u>	<u>OPERATION</u>
NOP	7000	<u>No operation.</u> This instruction causes a delay in program execution.
IAC	7001	<u>Increment AC.</u> The content of the AC is incremented by 1.
RAL	7004	<u>Rotate AC left.</u> The content of AC 1-11 is shifted into AC 0-10. The content of AC is shifted into the Link, and the content of the Link shifted into AC 11.
RTL	7006	<u>Rotate two left.</u> Equivalent to two consecutive RAL operations.
RAR	7010	<u>Rotate AC right.</u> The content of AC 0-10 is shifted into AC 1-11. The content of the Link is shifted into AC; and the content of AC 11 is shifted into the Link.
RTR	7012	<u>Rotate two right.</u> Equivalent to two consecutive RAR operations.
CML	7020	<u>Complement Link.</u> The content of the Link is complemented.
CMA	7040	<u>Complement AC.</u> The content of each bit of the A is complemented.
CLL	7100	<u>Clear Link.</u> The Link is loaded with a binary 0.
CLA	7200	<u>Clear AC.</u> Each bit of the AC is loaded with a binary 0.

Table VI (Cont.)  
Register Reference Instructions

<u>MNEMONIC</u>	<u>OCTAL</u>	<u>OPERATION</u>
HLT	7402	<u>Halt.</u> Clears the Run flip-flop so that program execution stops.
OSR	7404	<u>Logical OR with Switch Register.</u> The content of the programmer's console Switch Register (SR) is combined with the content of the AC by a bit wise logical OR operation. The result is left in the AC and the original content of the AC is lost. The content of the SR is not affected.
SKP	7410	<u>Skip.</u> The content of the PC is incremented by 1, to skip the next sequential instruction.
SNL	7420	<u>Skip on non-zero Link.</u> The content of the Link is sampled. If the Link contains a 1, the content of the PC is incremented to skip the next sequential instruction. If the Link contains a 0, the next instruction is executed.
SZL	7430	<u>Skip on zero Link.</u> The content of the Link is sampled. If the Link contains a 0, the content of the PC is incremented to skip the next sequential instruction. If the Link contains a 1, the next instruction is executed.

Table VI (Cont)

## Register Reference Instructions

<u>MNEMONIC</u>	<u>OCTAL</u>	<u>OPERATION</u>
SZA	7440	<u>Skip on zero AC.</u> The content of each bit of the AC is sampled. If every bit contains a 0, the content of the PC is incremented to skip the next sequential instruction. If any bit contains a 1, the next instruction is executed.
SNA	7450	<u>Skip on non-zero AC.</u> The content of each bit of the AC is sampled. If any bit contains a 1, the content of the PC is incremented by 1 to skip the next sequential instruction. If every bit contains a 0, the next instruction is executed.
SMA	7500	<u>Skip on minus AC.</u> The content of AC is sampled. If AC contains a 1, indicating that the AC contains a negative two's complement number, the content of the PC is incremented to skip the next sequential instruction. If AC contains a 0, the next instruction is executed.
SPA	7510	<u>Skip on positive AC.</u> The content of AC is sampled. If AC contains a 0, indicating that the AC contains a positive two's complement number (or zero), the content of the PC is incremented to skip the next sequential instruction. If AC contains a 1, the next instruction is executed.

Table VI (Cont)

Register Reference Instructions

<u>MNEMONIC</u>	<u>OCAL</u>	<u>OPERATION</u>
XOR	7500	<u>Exclusive OR with Switch Register.</u> The contents of the SR is combined with the contents of the AC by a bit wise exclusive OR operation. The result is left in the AC and the original content of the AC is lost.

Table VII

## Input/Output Transfer Instructions

<u>MNEMONIC</u>	<u>CCF</u>	<u>OPERATION</u>
KCF	6030	Clear keyboard/reader flag without operating the device.
KSF	6031	Skip the next instruction if the keyboard/reader flag is a 1.
KCC	6032	Clear the AC and the keyboard/reader flag.
KRS	6034	Read a character from the keyboard/reader buffer. The keyboard/reader flag is set when the operation is completed.
KIE	6035	Enable the keyboard/reader to cause program interrupts if AC bit 11 is a 1. Disable the keyboard/reader from causing interrupts if AC bit 11 is a 0.
TFL	6040	Set the printer/punch flag.
TSF	6041	Skip the next instruction if the printer/punch flag is a 1.
TCF	6042	Clear the printer/punch flag.
TPC	6044	Load the contents of AC bit positions 5-11 into the printer/punch buffer and operate the printer punch. The printer/punch flag is set when the operation is completed.
TSK	6045	Skip the next sequential instruction if either the printer/punch interrupt request flag or the keyboard/reader interrupt request flag is set.

Command	Location	Contents
CMA	7572	2163
	7573	1745
	7574	0477
	7575	----

The instruction is CMA (complement accumulator) and four microinstruction locations are available to implement the instruction. Note that only three of the four allotted locations (7572, 7573, and 7574) were necessary to implement the instruction.

When the instruction CMA is to be executed, the MPC (micro-program Program Counter) jumps to location 7572 in memory. The contents of that location are fetched (fetch is discussed later) and the microinstructions 21 (enable accumulator output), and 63 (enable A register input) are carried out. The A register is an auxilliary register located in the General Purpose Arithmetic unit. There is a B register located there that is used for a similar purpose (for more information on these registers, see Ref 1:9). The next microword is fetched and decoded as 17 (enable accumulator input), and 45 (complement the A register). After the A register is complemented, it is put onto the bus. Since the accumulator input is enabled, the complement of the original contents of the accumulator are put into the accumulator. The final microword does a return to fetch (04), and no operation (77). A return to fetch causes 7500 to be loaded into the MPC so the machine branches back to the beginning of a fetch state. The fourth microword is not used to implement the CMA instruction.

All program instructions in Pedagog are entered as described above. This ability to define main program instructions using microcode allows Pedagog a wide range of possible instruction sets.

#### Micro States

Located on the front panel just to the left of the single-step switch are eight MICRO STATE indicators. The indicators are labeled H, A, B, C, D, E, F, and G, and are used to indicate the internal state of the machine. The H is listed first because this is a Halt state. When the machine is in the RUN mode, or the STEP mode, these eight states are sequentially used to control register transfers. When the CLR switch is pushed, the machine automatically returns to the Halt state and is ready for operation. If the STEP switch is pushed, the machine advances from the Halt state to the A state. The micro states continue to increment as the STEP switch is activated. When the indicator reaches state G, the next state will be the Halt state and the sequence will repeat.

The first four active states of Pedagog (A, B, C, and D) fetch microwords from memory. The exact sequence is shown below:

State A	Put the contents of the MPC into the MMAR
State B	Put the contents of the MMAR into the memory's address register

State C     Put the contents of the memory address just  
              loaded into the MMBR

State D     Put the contents of the MMBR into the MIR

Note that what these four states have done is to "fetch" a microword from memory and place it in the MIR to be decoded. In states E, F, and G, the instructions of the MIRD are carried out. The exact action will depend upon the instruction being decoded. Note that state G is longer than any of the other micro states. Its indicator will be illuminated during states E and F as well as state G.

#### Stepping Through a Sample Program

In this section a short program will be entered into Pedagog. The program will be stepped through to give the user a general feel for the register transfers inside the machine. The sample program will consist of one main program instruction, the HALT instruction. Due to the large number of micro-steps associated with each memory fetch, over 50 activations of the STEP switch will be made. Only the first several steps are explained in detail. After that, only steps with significant information will be explained. To begin, enter the first five locations of Table II and their contents into memory. This is the microcode used to fetch a main program instruction. Go to location 7620 in Table II and enter the contents into memory. This is the microcode to carry out the HALT instruction. Examine each of

the locations to insure the proper values are entered. Next, enter 7402 (the HALT instruction) into address 7401. Set 7401 into the SWITCH REGISTER and push LOAD ADDR. This sets the Program Counter to the address of the main program instruction (HALT).

Push the CLR switch to clear the machine and press the STEP switch once. By using the DISPLAY SELECTOR, note that the contents of the MPC have been transferred to the MMR. Set the DISPLAY SELECTOR on MMR. When the STEP switch is pushed again, nothing noticeable will occur. Memory is being read during this state. When the STEP switch is pushed again, note that the contents of location 7400 (0477) are transferred into the MMR. When the STEP switch is pushed again, the contents of the MMR (0477) are moved to the MIR. The contents of the MIR are decoded during the next three states. Since the microword will be decoded as a "jump to fetch" (see Table IV), place the DISPLAY SELECTOR on MPC and observe the address 7500 being placed into the MPC during the next three states. Exactly where the register transfer takes place during states E, F, and G depends upon the particular instruction. In this case, 7500 will be placed into the MPC during state E.

When the MICRO STATE indicator is back to the Halt state, a new microword fetch cycle is ready to begin. This time, however, location 7500 will be the address of the microword fetched from

memory. Cycle through the micro states and note that during states E, F, and G, the contents of the Program Counter are placed into the MAR. This is accomplished by the contents of address 7500, which are 0165. Also note that during state E, the contents of the MPC were incremented to 7501. This means that the next memory location accessed will be address 7501. Step through the micro states again and note that 3067 is fetched from memory and is decoded as a memory "read" of the location specified in the Program Counter. The PC holds the address of the main program instruction (the HALT instruction located at address 7401). The result of reading memory (7402, the HALT instruction) is placed in the MBR during states E, F, and G. When the micro states are cycled through again, the microword 1570 is decoded and during states E, F, and G, the contents of the MBR are placed into the Instruction Register. The contents of the IR (7402) are decoded by the Instruction Register Decoder, and during the next cycle of the micro states, the output of the MIRD is placed into the MPC. This input to the MPC is the address in memory of the microcode necessary to carry out the HALT instruction. To demonstrate that the machine does halt under software control, cycle the MICRO STATE indicator to the Halt state and push the RUN switch. The machine should run momentarily and stop with 7621 in the MPC. The MICRO STATE indicator should show state E. Note that it is possible to "step" through part of the program and finish the remainder of

the program in the RUN mode. This is only possible if the machine is in the Halt state prior to activating the RUN switch.

The number of steps in a complex program would discourage stepping through a program. However, using the combination of stepping and RUN as just described makes the stepping mode a convenient and valuable debugging tool, as well as an aid to learning.

### Running a Sample Program

In this section a program is provided for the user to gain familiarity with the machine and the instruction set. The program can be stepped, executed in the RUN mode, or a combination of both.

The program listed below is a simple program that multiplies two numbers ( $14_{10}$  by  $9_{10}$ ) using successive addition. The solution  $176_8$  will appear in the accumulator. A detailed discussing of the program is not given here, but can be found in Introduction to Programming by the Digital Equipment Corporation (Ref. 9). This book was written exclusively for the PDP-8 and gives a detailed description of programming using the PDP-8 instruction set. The program is listed below:

Location	Mnemonic	Contents	Comments
7410	CLA	7200	Clear accumulator
7411	CLL	7100	Clear Link
7412	TAD 023	1023	Set up tally counter
7413	CMA	7040	Complement accumulator
7414	IAC	7001	Increment accumulator
7415	DCA 025	3025	Count additions of 14

Location	Mnemonic	Contents	Comments
7416	TAD 024	1024	Add 14
7417	ISZ 025	2025	Skip if tally is zero
7420	JMP 016	5016	Add 14 unless done
7421	CLL	7100	Clear link
7422	HLT	7402	Halt after 9 adds
7423		0011	9
7424		0016	14
7425		0000	Holds tally

#### Microprogramming New Instructions

The microcode in Table III allows the Pedagog to emulate the PDP-8 instruction set. This section describes how an instruction set can be altered, or how a new instruction set can be developed and used on the Pedagog. As an example, the PDP-8 instruction "Increment and Skip if Zero" (ISZ) will be altered to perform "Decrement and Skip if Zero" (DSZ).

The ISZ instruction has the form 2XXX. XXX indicates an address. The main program instruction 2024 (ISZ) indicates that the contents of the memory location 0024 should be incremented. If after the contents are incremented the value is zero, the PC is incremented by 1 (the next sequential instruction is skipped). If the value does not become zero, the next instruction is executed. The content of location 0024 is one greater than before the instruction was executed. The microcode for the ISZ instruction is listed below:

2164	(Put contents of AC into B register)
2474	(Put the address part of the MBR into the MAR)
3063	(Read memory and put the contents in the A register)

3672	(Increment A and put it into the AC)
1077	(If AC = 0, increment the PC)
2167	(Put contents of AC into the MBR)
0577	(Load what is in the MBR into memory)
0772	(Put contents of B register into AC)
0477	(Put 7500 into MPC, or return to fetch)

Note in the above microcode that the first instruction, 2164, temporarily stores the contents of the accumulator (AC) in the B register. The accumulator must be used during the remainder of the instruction, and the contents of the accumulator would be lost if they were not first stored. Just before the "return to fetch" the contents of the B register are returned to the accumulator. The remainder of the microcode is self-explanatory.

To convert the ISZ instruction to a DSZ instruction involves only a minor change. Since it is still desired to have the instruction "skip if zero", the only change necessary would be microword four (3672). Rather than increment the register A and put the result into the accumulator, the instruction should be changed to decrement the A register and put the result into the accumulator. From Table IV it can be seen that the microword 3672 used in the ISZ instruction should be changed to 1172. This is the only change necessary to convert from ISZ to DSZ.

Using the technique discussed above, instruction sets can be altered or added with little difficulty. Custom made

instructions can be tailored to meet program needs. For example, an instruction such as "XOR SWITCH REGISTER with AC, Shift one left, and increment PC if Link = 0", could be implemented as shown below:

2163	(Evoke SR output, Evoke B register input)
3164	(Evoke AC output, Evoke A register input)
1753	(Evoke AC input, Evoke exclusive OR)
7771	(No operation, Shift AC left)
0677	(If Link = 0, Increment PC, No operation)
0477	(Return to fetch)

The variety of instructions is limited only by the programmer's imagination and ingenuity. When a new instruction is written, the single-step feature of the Pedagog can be used to insure the instruction performs as intended.

### Summary

This user's manual has described the various aspects of programming the Pedagog II computer. A description of each front panel feature was given along with a discussion of the machine memory.

Microprogramming main program instructions was discussed along with microcode descriptions and definitions.

Two sample programs were presented. One program demonstrated the single-step function of the computer. The other program demonstrated the main level program instructions of the PDP-8 instruction set.

The last section described how to alter the instruction set used in Pedagog. One example demonstrated how the instruction "Increment and Skip if Zero" could be changed to "Decrement and Skip if Zero". The other example described how to tailor an instruction to meet a specific program need.

### Vita

James Stephen Sheehan was born on 26 March, 1950 in Tulsa, Oklahoma. He graduated from high school in 1968 and later attended Oklahoma State University from which he received the degree of Bachelor of Science, Electrical Engineering in June 1973. Upon graduation he received a commission in the United States Air Force through the Reserve Officers Training Corps. He then attended Undergraduate Pilot Training at Williams AFB, Arizona. Later he served as a T-38 Instructor Pilot at Reese AFB, Texas until entering the School of Engineering, Air Force Institute of Technology, in June 1978.

Permanent address: 1802 Oak Street  
Collinsville, Okla.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GE/EE/79-33	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) PEDAGOG II REALIZATION		5. TYPE OF REPORT & PERIOD COVERED MS Thesis
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) James S. Sheehan Capt, USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT-EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE December 1979
		13. NUMBER OF PAGES 257
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES Approved for public release; IAW AFR 190-17 J. P. Hipps, Major, USAF Director of Public Affairs		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Microprogrammable computer Educational computer PDP-8 Minicomputer emulator		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) A small microprogrammable computer, designed in a previous effort, was evaluated to determine its condition. Found non-operational, it was analyzed to determine areas that required design changes. Several modules of the system were redesigned. The system was tested to insure proper operation. A new control panel and cabinet were designed and constructed. The computer backplane was installed in a		

DD FORM 1473

EDITION OF 1 NOV 65 IS OBSOLETE

458

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

cabinet. Existing power supplies were adapted for use, and another was designed and added.

A bus interface, designed in a previous effort to allow expansion of memory, was fabricated and tested. It was found non-operational.

The machine is microprogrammable and has a 256-word, random access memory. Microcode was written to allow the machine to emulate the 36 basic instructions of the Digital Equipment Corporation (DEC) PDP-8. All major register transfers can be seen on the front panel. The machine may be taken to the classroom for instruction or demonstrations in basic microprogramming.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)